UNIVERSITY OF TARTU
FACULTY OF SCIENCE AND TECHNOLOGY
Institute of Computer Science
Software Engineering Curriculum

Sunday Ayandokun

Application-agnostic Personal Storage for Linked Data

Master's Thesis (30 ECTS)

Supervisor: Peep Küngas, PhD

Tartu 2016

Application-agnostic Personal Storage for Linked Data

Abstract:

Recent advances in cloud-based applications and services have led to the continuous replacement of traditional desktop applications with corresponding SaaS solutions. These cloud applications are provided by different service providers, and typically manage identity and personal data, such as user's contact details, of its users by its own means.

As a result, the identities and personal data of users have been spread over different applications and servers, each capturing a partial snapshot of user data at certain time moment. This, however, has made maintenance of personal data for service providers difficult and resource-consuming. Furthermore, such kind of data segregation has the overall negative effect on the user experience of end-users who need to repeatedly re-enter and maintain in parallel the same data to gain the maximum benefit out of their applications. Finally, from an integration point of view – sealing of user data has led to the adoption of point-to-point integration models between service providers, which limits the evolution of application ecosystems compared to the models with content aggregators and brokers.

In this thesis, we will develop an application-agnostic personal storage, which allows sharing user data among applications. This will be achieved by extending AppScale app store identity infrastructure with a personal data storage, which can be easily accessed by any application in the cloud and it will be under the control of a user. Usability of data is leveraged via adoption of linked data principles.

Keywords: Appscale, personal data storage, linked data.

CERCS: P170

Personaalne andmeruum lingitud andmetele

Lühikokkuvõte

Personaalsete andmete ristkasutuse puudumine veebirakenduste vahel on viinud olukorrani kus kasutajate identiteet ja andmed on hajutatud eri teenusepakkujate vahel. Sellest tulenevalt on suuremad teenusepakkujad, kel on rohkem teenuseid ja kasutajaid, väiksematega võrreldes eelisseisus kasutajate andmete pealt lisandväärtuse, sh analüütika, pakkumise seisukohast. Lisaks on sellisel andmete eraldamisel negatiivne mõju lõppkasutajatele, kellel on vaja sarnaseid andmeid korduvalt esitada või uuendada eri teenusepakkujate juures vaid selleks, et kasutada teenust maksimaalselt.

Käesolevas töös kirjeldatakse personaalse andmeruumi disaini ja realisatsiooni, mis lihtsustab andmete jagamist rakenduste vahel. Lahenduses kasutatakse AppScale rakendusemootori identiteedi infrastruktuuri, millele lisatakse personaalse andmeruumi teenus, millele ligipääsu saab hallata kasutaja ise. Andmeruumi kasutatavus eri kasutuslugude jaoks tagatakse läbi linkandmete põhimõtete rakendamise.

Võtmesõnad: Appscale, personaalne andmeruum, linkandmed.

CERCS: P170

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank almighty God for given me the grace to start and finish this study. Secondly, I appreciate my parents, my entire family and my friends for their full support and prayers during this journey. I also want to thank all my wonderful Professors, Associate Professors and Technical Assistants who instilled in me the right knowledge and attitude. I am so grateful to my supervisor, Peep Küngas, Ph.D. – a great researcher, for his immense intellectual contributions to the completion of this thesis.

In addition, I want to thank my colleagues in the office at Zero Technologies for sharing their thoughts when called upon. A big thank you to the Estonian government, University of Tartu, Tallinn Technical University and IT Academy for providing an enabling environment for learning. Last but not the least, to my fiancée – Temitope Adenuga, thank you for your prayers and encouragements all through.

1. Introduction

In today's emerging IT technologies, cloud computing [1] has played a vital role in unlocking great IT innovations, as organizations have seen its adoption have brought unprecedented growth in the way they go about their business operations. This is obvious, as it offers a promising paradigm that could enable businesses to face market volatility in an agile and cost-efficient manner. [1] Its adoption has helped in reducing costs, offering tremendous flexibility, reliability and enabling processing of massive amount of data on commodity hardware. It has also helped organizations to open their services to a large number of customers with little or no geographical limitations. Hence, the cloud presents an undeniable potential to benefit all users and businesses.

Cloud computing, shortly referred to as cloud, is an on-demand computing model which enables access to computing resources such as services, applications, networks, servers, and storage. The cloud enables rapid provisioning and release of these resources with reduced effort and less service provider interaction. [2] The main principle behind this model is referred to as offering computing, storage, and software "as a service". It delivers computing as a utility, a business model where users of computing resources pay providers based on usage ("what-you-used-is-what-you-pay-for"). As promising as the cloud is, it comes with its challenges. User identity and personal data management are difficult due to the significant dependencies between several services connected in the cloud [3]. Users' data portability across domains and different cloud applications is also a problem that needs a solution in order to unlock cloud full adoption.

In today's social networking, we have seen a rise of social networking sites such as Facebook, Google, and LinkedIn becoming identity providers and personal profile data managers. Moreover, from the application development point of view app stores such as Google Play and Samsung Apps have become environments, which take care of identity management such that application developers can focus on improving their apps. In addition, the app stores take care of common tasks such as application distribution, billing and user management. In this thesis, as a proof of concept, we will make use of an existing open source app store similar to those mentioned above called AppScale. It is an open source implementation of Google App Engine (GAE). It is API-compatible with GAE and thus executes GAE applications without modification. More details will be provided in chapter 4.

1.1 Problem Statement

Cloud-Scale Identity Fabric like those mentioned above should enable the transfer of user data across application domains, that is if application X is where I primarily store my personal data, then I should be able to delegate application Y to fetch my data from X securely. Such Fabric should also be able to provide features such as:

- Access control and authorization,
- Federation Authentication and Single sign-on (SSO),
- User account management and Provisioning
- Auditing and Compliance
- Cloud-based scalability, Regulations,

The above listed must evolve, in order to realize a cloud-scale identity fabric [3].

As mentioned earlier, user identities and personal data of the same user are distributed between different applications and servers since cloud users have to fill in their personal data every time they are about subscribing to use a particular cloud-based application. Hence, this does not give individuals control over their data since there is no single point of personal data storage for easy management and control.
With such constraint, user data portability across different cloud applications and domains has not been fully implemented, since every application does user data management differently and internal to the application; user management interfaces are neither consistent nor standardized [4].

Also, from data privacy point of view, as there has been serious awareness on user data privacy in recent time, more and more people are becoming unwilling to release their data during signing up to new cloud applications, as people are having a new understanding of personal data which is an economic asset generated by the identities and behaviors of individuals while engaging with IT services.

We believe that user's unwillingness to release data pose a great challenge to emerging IT services that could benefit businesses, individuals and the world at large. The privacy problems worth mentioning are; users are not aware of the usage of their data when released to service providers [5]. In [6], they suggested service providers should develop a scenario in which the user actually understands what will happen to their data, getting to a point where providers of cloud apps will explain clearly, concisely, and very simply to the user what is

happening with their personal data. In addition, having a good understanding of what 'personal data' means is also essential to both users and service providers. According to EU Data Protection Directive, 'Personal Data' is any information relating to an identifiable natural person; "*an identifiable person is one who can be identified, directly or indirectly, in particular by reference to an identification number or to one or more factors specific to his physical, physiological, mental, economic, cultural or social identity*" [7].

Interestingly, the concept of personal data ecosystem (PDE) has been proposed [8]. It is an emerging intellectual activity of companies and organizations that believe individuals should be in control of their personal information and directly benefit from its use, making available a growing number of tools and technologies to enable such control [8]. The PDE is expected to address the privacy challenges in personal data lifecycle – Data Harvesting, Data Mining and Application [8] [9]. More specifically, a new EU act will be introduced, which will enforce companies to provide on-demand access to personal data, if the person asks for it [9]. Towards this end, we have highlighted key research questions that this thesis aims to address:

1. How can we fully represent personal user data which is self-contained as personal data storage, one that is standard and not application-specific?
2. How will multiple cloud applications link to a single user identity and personal data, one that is not constrained to the data model of any application?
3. But with so many applications coming from different providers, how will end users' personal data be accessed securely with user's authorization, to a specific section of their personal data requested by the third-party application?

1.2 Goals Of The Thesis

Based on the identified challenges of personal data been constrained to a specific application data model, in addition to the fact that users don't have control over their data, we have identified a possible solution. From data model perspective, the linked data concept and Resource Description Framework (RDF) seems a suitable candidate for a representation, which is not application specific.

This thesis aims to design and implement a model for an existing app store for application-agnostic personal storage by means of linked data. This will be achieved by extending AppScale [10] - an open source implementation of Google App Engine. We will validate the personal data storage implementation by using it as a user management platform for the personal data storage for inforegister.ee.

1.3 Organization Of Thesis

Chapter 2 discusses related works. In Chapter 3, we cover the background of selected candidate solutions for the implementation i.e. Linked Data and Resource Description Framework. In Chapter 4, we describe the details of the design, architecture and major components of the proposed solution. Chapter 5 provides detailed proof of concept implementation with inforegister.ee data and functionality. Chapter 6 presents the validation of the solution for inforegister.ee with a realistic scenario; performance measurements under different constraints/scenarios. Chapter 7 concludes the thesis and discusses potential directions for future work.

2. Related Work

In this chapter, we present related works of implementation of personal data storage in the cloud. We will review the implementation, techniques, and architecture of the existing similar solution. The rise in the accumulation of user data in the cloud by service providers has seen a critical demand for solutions that can help users manage and have control over their data that is being collected. There are existing implementations generally referred to as Personal Data Service (PDS), which can offer users the desired control on their data.

PDS, as defined in Wikipedia, is a personal digital identity management service which is controlled by an individual. It gives users a single point of control for their personal information [11]. Such pieces of information are stored in external distributed repositories which can be accessed via an application programming interface (API). The user can permit and revoke access to their data from third party requester. Generally, PDS empowers the user to be in control of their data, with the ability to manage personal information; have a dashboard view of their online behaviors and activities; provides identity and claims verification [12]; also to be able to share a section of their data with the organization of one's choice based on the specific data section that the organization require instead of an absolute access to all user data.

As opposed to how user's data are collected and managed by different applications and service providers, [12] considered the current model as inefficient and broken due to the following reasons:

- Users have limited control over the usage and management of their data;
- Disparity in privacy and terms of usage by each service;
- Users shared too much detailed personal information than required, which increases chances of exposure;
- The various services have a partial view of each user which leads to error.

In view of the above, we can see the danger posed by the status quo on every single cloud users as regards the privacy and usage of their personal data as they interact with various cloud services.

The following are the various cloud-based PDSes reviewed, with the potential to address the major issues highlighted above.

2.1 Personal.com

Personal provides the Web and mobile service to give users a data vault and tools to control, share and gain value from their personal information, including through personal networks [12]. Personal as a commercial Personal Data Storage with a centralized attribute data store which house user data, allows individuals to add or import their data and share it through bundles of structured and unstructured fields of data and files called Gems. It also provides an API to enable bi-directional attribute updates from third-party web services. Personal as a user data management service allows people to work securely with others to organize and use the information that powers their life [13].

The service works such that, users register on the platform and start filling their personal information into different identified categories.

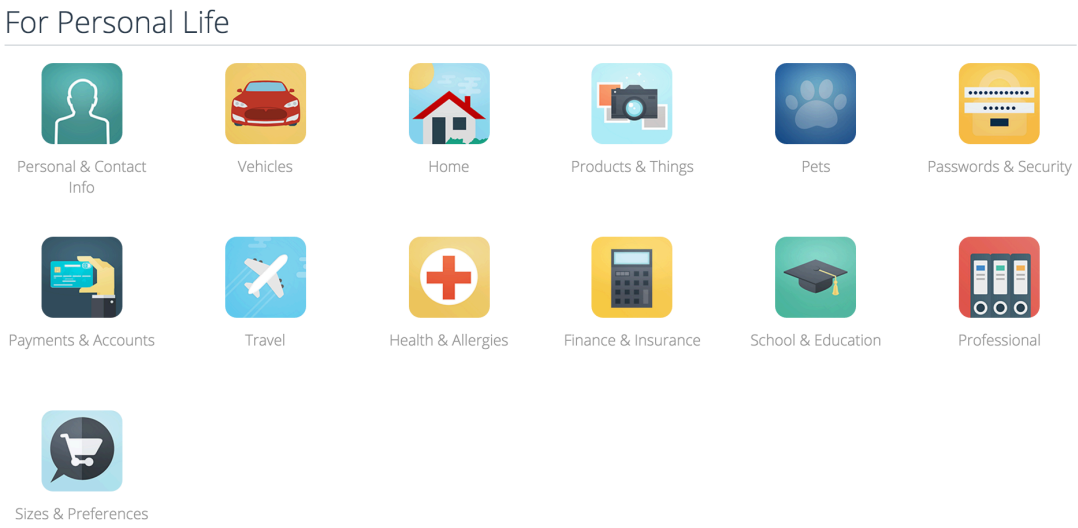

Figure 1: Personal user data categories[1]

[1] https://www.personal.com/apps/home/ - /cat/personalapps (visible to only registered users)

The user can grant access to their information and also revoke the access when no longer require. The personal has a free version and premium version (https://www.personal.com/tour/pricing/).

Being a commercial solution, there isn't much description of the underlying architecture. But as described in [12] as a case study, Personal Platform, has a Privacy by Design architecture and offers a full suite of APIs to support for various types of functionality around data management.

2.2 ID Hole

This is another commercial consumer PDS, it allows users store personal data and also provides mean of sharing such data with other parties.
ID Hole[2] provides all types of users, such as businesses, organizations, professionals, students, and all others who frequently utilize the internet with the opportunity to save personal and/or business information.

Similar to personal.com users fill in their information they would like to store on ID hole. But unlike Personal.com, ID Hole users create their various data categories by themselves. ID Hole.com provides users with both a password and the encryption key to access it. This encryption key is the master access key to the user data and only the user has access to it if users forget this, access to the user account is lost forever. [14].

ID hole is just storing the user data on their server for the user access only and there are no means for users to delegate access to such information via any means. This approach of storing user data is quite different from our own approach as we intend storing the user data independent of application data structure using Resource description framework (RDF) which can also be shared.

[2] https://www.idhole.com/

2.3 Ownyourinfo.com

Similar to personal.com, Ownyourinfo.com also provides users with a personal data store, where users enter their data and organized in different categories. The stored data can be shared with someone else. This solution also lacks dynamic access and sharing of data with cloud services, as it presents a person-to-person data sharing model. Users have 5 free sharing per month with a premium version providing unlimited sharing.

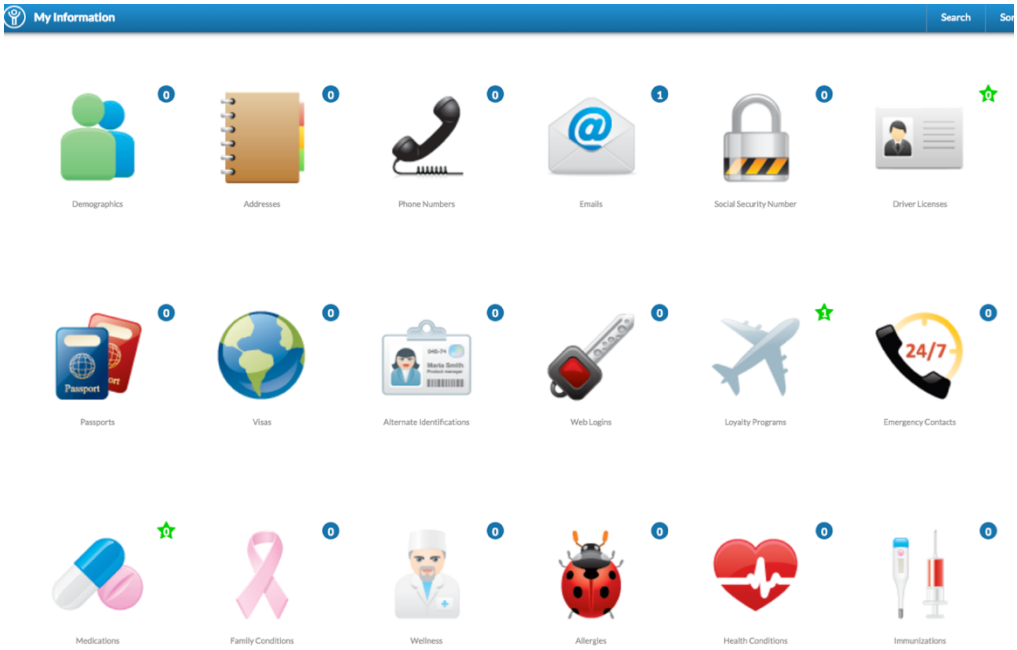

Figure 2: Ownyourdata user's information category[3]

2.4 The Locker Project

A Locker is a container for personal data, it gives data owner the ability to control how their data is protected and shared. It works by retrieving and consolidating data from different sources, to create a single collection of the things users see and do online like the places users visit, the links they share, contact details for the people they communicate with etc. It is an open source, JavaScript-based, PDS with a centralized underlying attribute store that exists on a person's personal computer as well as an API to support local applications [15]. It also provides APIs for developers to build applications with access to users' information.

The project is not in active development according to the project's Github page[4], but its developer changed the focus of development to Hallway[5], a multi-tenant version of the

[3] https://app.ownyourinfo.com/-!/profiles/667/categories/all(required registered user access)

Locker empowering Personal Data Application. Hallway project helps developers build applications which aggregate data easily from different service providers via one API [16].

2.5 OpenPDS

OpenPDS is an open-source Personal Data Store, enabling users to aggregate, keep, and grant access to their data while protecting their privacy. The system ensures that most processing of sensitive personal data takes place within the user's Personal Data Store [17]. This is achieved via an innovative framework for installing third-party applications. In the researchers' view, with the amount of data sources that a user interacts with daily, data exchange among different services is not enough. Rather, there is a need for users to have their own protected space, a Personal Data Store (PDS) acting as a single point where his/her data is stored. With the PDS, users can control who can access their data and manage authorizations for accessing the data. [18]

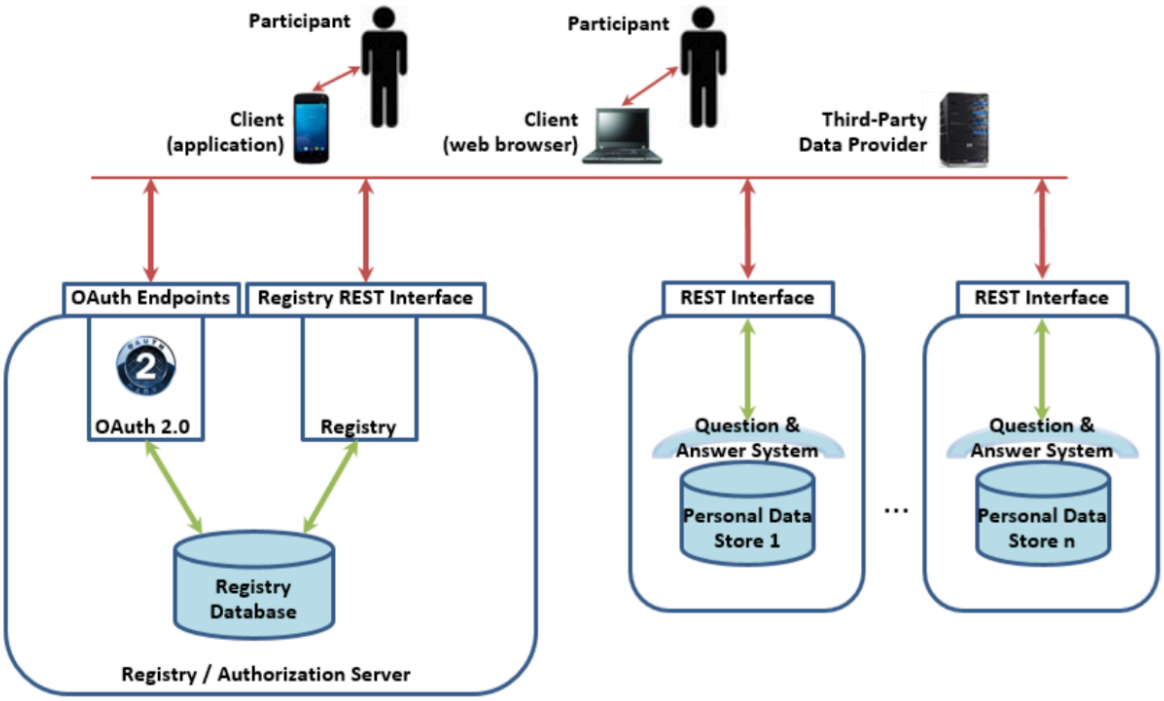

Figure 3: High-Level personal data ecosystem.[6]

Figure 3 shows the high-level personal data ecosystem of OpenPDS.

[4] https://github.com/LockerProject/Locker
[5] https://github.com/Singly/hallway
[6] github.com/HumanDynamics/openPDS/blob/master/doc/openPDS%20Developer%20Documentation.pdf

The Registry creates a profile for the user at the point of user registration and a personal data store is initialized for the user. The authorization server provides secure user authentication and authorization of access to personal data stores. The OAuth 2.0 protocol [19], the component is tightly coupled with registry providing account management services.
The distributed Personal data stores design shows a user-centric design, in which a single OpenPDS server supports each end user having separate backend database, user-specified encryption keys for all personal data in the data store.

In order to support such arbitrary schema, the researchers have chosen MongoDB[7] as the primary backend storage system. The OpenPDS implementation seems a close approach to our proposed implementation of personal data storage. In this view, our implementation is going to be built upon some of the relevant concepts of OpenPDS.

2.6 OPENi Personal Cloudlet

Personal Cloudlet is part of the OPENi EU FP7[8] funded project, with the aim of providing a platform that offers users, flexible control over their personal data.
This research project focuses on promoting innovation in the European mobile applications industry and they aim to achieve this by developing an open-source platform for consumer-centric mobile cloud applications [20]. The central concept is to minimize the scattered and duplicated users' data across various cloud services.

OPENi provides application users with a single point of data storage and control. This will enable consumers to manage what section of their data is available to each application and for what purpose. They believe this can serve as a single authoritative source for the consumers' personal data and content [20].

The aim of this project as highlighted in the project objectives[9] aligns well with our work in this thesis. We present below a brief description of OPENi architecture and some other concepts as it relates to our work.

[7] https://www.mongodb.org/
[8] https://ec.europa.eu/research/fp7/index_en.cfm
[9] http://www.openi-ict.eu/objectives/

2.6.1 OPENi's Architecture

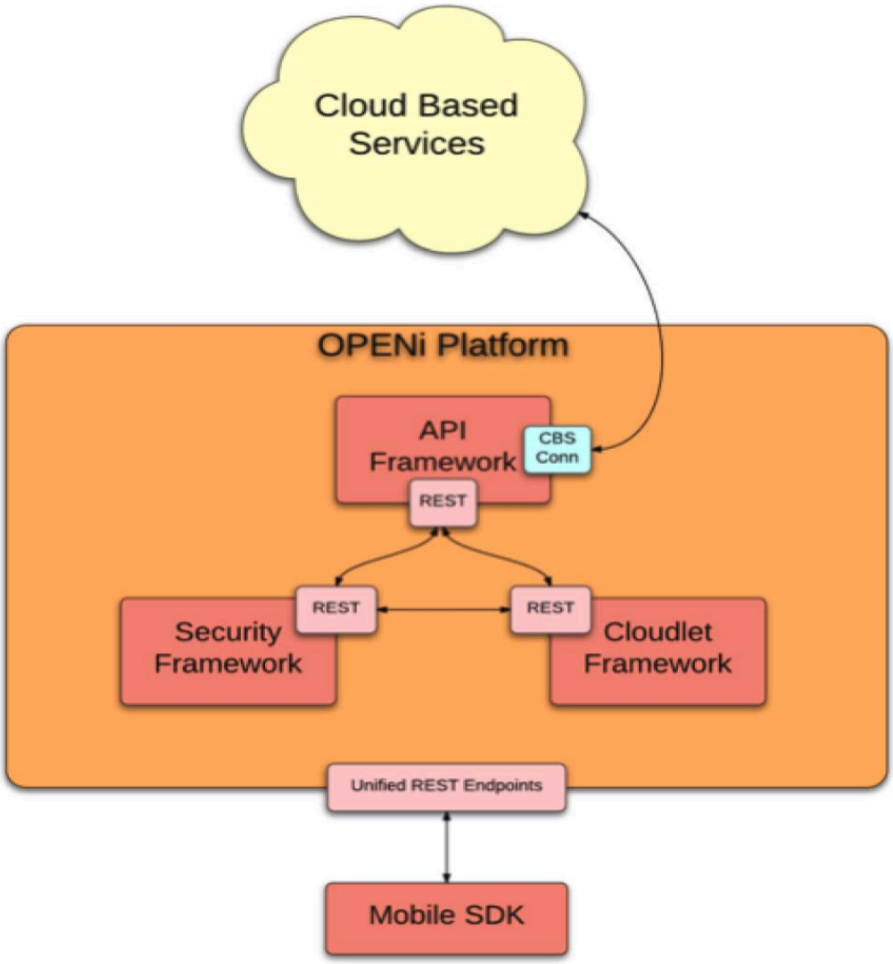

Figure 4: OPENi platform's high-level architecture *[21]*

The mobile SDK abstracts and simplifies access to OPENi services across multiple mobile platforms with a design that promotes rapid application development for easy developer onboarding. [21] The security framework implements the access control functionality which allows users to really have total control of their data. The API framework is the OPENi Graph framework which is an open-source framework capable of interoperating with a variety of cloud-based services and the detailed description of it can be found here [22]. The last core component is the Personal cloudlet framework [20] which provides application consumers with a single location to store and control their personal data, the feature which is achieved by a collaboration with the security framework.

2.6.2 OPENi User-Centric And Privacy-Preserving Features

OPENi uses the various technologies to achieve the core feature of privacy preserving and user-centric: It implements the OAuth 2.0 compliant flow for User Authorization which presents the user with a login view for authentication and a permission dialog for granting access to third-party apps. In order to maintain the framework statelessness, OPENi enhanced JSON Web Tokens (JWT[10]) which are digitally signed base64 encoded JSON objects that enable stateless REST based frameworks manage sessions and claims. [21]

2.6.3 Other Cloudlet Concepts

Cloudlet platform uses Couchbase which is a NoSQL Datastore as its backend data storage. This was driven by a requirement for a platform that is scalable. Cloudlet implemented RESTful object-based access to enable users share and control access to their information**.** As a promise to focus on users' privacy and control, Personal Cloudlet Framework's has some key features to achieve this which include: The Privacy Preserving Data Aggregator, the fine-grained access control and User Dashboard [20].

The OPENi project also carried out a detailed security analysis of Cloud-based services and OPENi Cloudlets in [23]. They highlighted some threats and solutions around data security as presented below:

Threat area	**Solution**
Data-in-transit	Use of secure protocols
Data-at-rest	Encryption, data tagging
Process / multi-tenancy	Data tagging
Data remnants	Clearing, sanitization, high-level SLA

Table 1: Data security issues and solutions[11]

[10] https://jwt.io/

[11] http://www.openi-ict.eu/wp-content/uploads/2013/11/OPENi_D2.3.pdf

2.7 MyDex Personal Data Store

MyDex is another Software as a Service Persona Data Store similar to Persona.com. It promises to allow users exchange their data with confidence.

In their white paper [24], they highlight what a personal data store initiative means to all the stakeholders. For individuals, it offers benefits such as convenience, Insight, emotional benefit of empowerment etc., also, as oppose to organization-centric approach PDS brings notable benefits to organizations, such as data accuracy and quality, Data completeness and richness. These and much more tend to guide MyDex to deliver values to the PDS users.

MyDex users can use the service free of charge, and MyDex only makes money when a user share data with a paying third-party. It is built using various open source components such as Vagrant[12] for the development environment, Git[13] for version control, GPG[14], OpenSSL[15] for Encryption, Percona Server[16] as the database, Symfony[17] as the platform framework etc.; further details can be found here [25]. During sign-up a user creates a private key for data encryption, which the user will need to provide after every login to decrypt user's PDS; this gives only the user access to the data, not even MyDex can access user data. It also allows users to connect to any organization of their choice in which they can share their data with and also receive data update from such organization.

MyDex also provides a mydex-browser-extension with features such as bookmarks management, browsing history, credentials management with auto-fill support. Users can add it as a connection to their data with a set of selected permissions.

In order to enable users to have more control over their data, MyDex provides a standard data sharing agreement which third-party service providers must agree to. [26]

[12] https://www.vagrantup.com/
[13] http://git-scm.com/
[14] https://www.gnupg.org/
[15] https://www.openssl.org/
[16] http://www.percona.com/software/percona-server
[17] http://symfony.com/

2.8 Comparative Summary

This section presents a summary of the reviewed related works; we have presented below a comparison table of the various PDS solutions with selected comparison parameters:

Features\PDSes	Personal	ID Hole	Ownyourinfo	Locker Project	OpenPDS	Personal Cloudlet	MyDex
Open source	No	No	No	Yes	Yes	Yes	No
Hybrid (Open/Commercial)	No	-	-	-	-	Yes	No
Data sharing model	P2P, P2B	-	P2P	P2P, P2B	P2B	P2B, P2P	P2B, P2P
User access control	Yes	-	Yes	Yes	Yes	Yes	Yes
Predefined vocabularies/data categories	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Internal Storage	-	RDBMS	RDBMS	-	NoSQL	NoSQL	RDBMS
Data multi-tenancy model	Yes	-	-	Yes	No		-
Right to be forgotten	Yes	Yes	Yes	-	-	-	-
Data Portability	Yes	No		-	-	Yes	Yes
Active Development	Yes	No	Yes	No	Yes	Yes	Yes

Table 2: Comparison summary table for the reviewed related works

Note: '-': Information Not available, **P2P**: Peer-to-Peer, **P2B**: Person-to-Business

- **Open Source** - this tells if the project is open-source
- **Hybrid** – available both in open-source and commercial
- **Data sharing model** – if it allows Peer-to-Peer, Person-to-Business (sharing the data with third-party service providers)
- **User Access control** – if it provides user access control on personal data, e.g. on a granular level
- **Predefined vocabularies** – whether there are set of defined data categorize or the user can create.
- **Internal Storage** – the choice of Datastore for the backend
- **Data multi-tenancy model**[18] – whether users' data are stored in a shared database

[18] https://msdn.microsoft.com/en-us/library/aa479086.aspx

- **Right to be forgotten** – it tells if user can delete their account and data completely on the platform
- **Data portability** – can user move their data to other PDS platform
- **Active Development** – if the project is still on-going

We are also aware of other attempts at implementing a personal data store, but which have seen little or no adoption. Projects such as AllAdvantage[19], Lumeria, infomediary, Bynamite[20], most of which are no longer in existence.

2.9 Privacy Issues And The PDS

We have seen how the ideas behind various PDS solutions tends to the protection of privacy, nevertheless, cautions must be taken to ensure that users' personal information is truly safe. In [12] the author raises some concern such as interoperability, interactions and information-sharing mechanisms between PDS stakeholders; that may affect privacy. It has been established that individuals are in control of their data. But what happens when such data is shared with the wrong party – supposed trust party. According to [12], taking a proactive approach will be crucial to the success of any PDS initiatives; which can be achieved via Privacy by Design., transparency and clarity will be essential.

In addition, PDS service providers must develop easy-to-use features, ensure granular data sharing, privacy-protective protocols, facilitate interoperability between data sets and also sensitize users of the privacy inference of his or her data sharing decisions.[12]

19 http://www.alladvantage.com/
20 https://www.crunchbase.com/organization/bynamite

3. Background

This chapter focuses on an introduction to the various technologies that will be used in our proposed solution. Such as the semantic web powered by concepts like Linked Data[21], Resource Description Framework(RDF[22]), Virtuoso[23], Appscale[24] – the scalable Application Platform as a Service, Django – Python web framework, OAuth 2.0[25], Tyk[26]. Also to be discussed are the reasons for chosen these technologies.

3.1 Linked Data

As discussed previously and in most of the reviewed works, the huge growth in user data generated daily as users interact with cloud services has raised concerns among various stakeholders ranging from the industry and academics. One can imagine this huge amount of data and wonder how applications can utilize this in a constructive and valuable way. To be easy for innovative application usage of such data, the data must be machine-readable and also enable linkages among related data.

Linked Data concept is about harnessing the Web to bring together unlinked related data. Concisely, Linked Data defined in [27], "*refers to a set of best practices for publishing and connecting structured data on the Web.*" Such data are machine-readable, self-descriptive, linked to data from external sources and vice-versa. Linked Data depends on documents consisting data in RDF format, which allows making typed statements that connect arbitrary things in the world and results to what is referred to as the Web of Data. [27]

The huge amount of data generated daily on the web would make more sense and ease sharing difficulty across several applications if linked data concepts are properly applied. Just as we can see in W3C[27] definition, Semantic Web often referred to as the **web of data** "*provides a common framework which permits data to be shared and reused across application, enterprise, and community boundaries. It is based on the Resource Description Framework (RDF[28])".* [28]

[21] http://linkeddata.org/
[22] https://www.w3.org/RDF/
[23] http://virtuoso.openlinksw.com/
[24] https://www.appscale.com/
[25] https://tools.ietf.org/html/draft-ietf-oauth-v2-17
[26] https://tyk.io/
[27] https://www.w3.org/
[28] https://www.w3.org/RDF/

It's inevitable to think that trying to link data from different sources, both structured and unstructured could turn out to be messy. In other to avoid that, one must follow the proposed set of rules[29] by Sir Tim Berners-Lee - the father of the current World Wide Web. The rules commonly known as the 'Linked Data principles' include:

1. URIs should be used as identifiers for things,
2. HTTP URIs should be used so that people can find out more about those things
3. Give useful information, using the standards (RDF, SPARQL) when someone accessed a URI,
4. Add links to other URIs, to discover more things

These rules guide publishing data on the web in a way that the linked data from different sources becomes part of a single global data space. [27]

Our choice of Linked Data approach as the data model for personal data storage is informed by the opportunities in Linked data. In [29], we have seen a lot of benefits of Linked Data which contributes to our choice. Some of which includes:

- It is applicable to structured, semi-structured, and unstructured data
- Elimination of internal data locked down in 'silos',
- Ability to integrate both internal and external data,
- Inter-linkage of enterprise, industry-standard, open public data,
- Robust data modeling is provided for any legacy schema,
- Adaptable and painless updates to existing schema

We believe Linked Data has a lot to offer application developers for a better interoperability due to its essential characteristics. With Linked Data, resources are self-descriptive, good separation of concern between formatting and presentation. "*The use of HTTP standardized access mechanism and RDF as a standardized data model simplifies data access compared to Web APIs, which rely on the heterogeneous data model and access interfaces."* [27]

More so, the linked data approach answers one of our research questions - **"*How can we fully represent personal user data which is self-contained as personal data storage, one that is standard and not application specific?*"**

[29] https://www.w3.org/DesignIssues/LinkedData.html

3.2 Resource Description Framework (RDF)

RDF is a standard directed graph-based data model for data exchange on the web [30]. RDF creates one of the essential units for forming a web of semantic data. It consists of a subject, predicate and object called triples. The predicate provides the linkage between the subject and object. RDF has features that enable combining several data even if the underlying schemas are different, hence, the model allows both structured and semi-structured data to interoperate, and can be easily shared across different applications.

The RDF model encodes data as subject, predicate, object triples. The subject and object of a triple identify a resource, or a URI and a string literal respectively. The predicate tells how the subject and object are related, and it is also represented by a URI. [27] For example, a triple can be used to relate this thesis and the author. Thesis and author are the subject and object respectively. And are related as Thesis 'written by' Author.

The example can be encoded in RDF triples link as:

<http://resources/thesis/1> <http://examplevocabulary/writtenBy> <http://persons/authors/author_identifier> .

Subject: http://resources/thesis/1

Predicate: http://examplevocabulary/writtenBy

Object: http://persons/authors/author_identifier

In view of the above, this data model gives linked data an edge having the advantage of being interoperable with other data set and being machine-readable due to the semantic graph structure. Further description of RDF could be found here [31].

3.3 Vocabularies/Ontologies

In order to have information described in a commonly understood way and unambiguously interpreted, there must be a set of standard vocabularies to describe things in different domains. *"These vocabularies can be reused by various data producers when describing data about a given subject, making such data semantically interoperable."* [32] In this thesis we used the FOAF[30], SIOC[31], Vcard[32], Public Procurement[33]. The choice of these ontologies is informed by the domain of the test data we will be working with.

[30] http://xmlns.com/foaf/spec/

[31] http://www.sioc-project.org/

[32] https://www.w3.org/TR/vcard-rdf/

[33] http://purl.org/procurement/public-contracts - Contract

It is important to be aware of existing vocabularies when describing things in a particular application domain. There exist some semantic search engines to find out the most appropriate vocabulary for a domain. Some of which include: Swoogle[34], Linked Open Vocabularies[35], DCMI Metadata Terms[36]. In the next section, we will describe briefly some of the popular ontologies.

3.3.1 Friend-Of-A-Friend (FOAF)

FOAF[37] is mainly used to describe social networks of human collaboration, friendship and association and includes core classes such as Agent, Person, name, title, familyName, givenName, knows, member etc.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:foaf="http://xmlns.com/foaf/0.1/">
    <foaf:Person rdf:about="http://about.me#sunday"
xmlns:foaf="http://xmlns.com/foaf/0.1/">
        <foaf:name>Sunday Ayandokun</foaf:name>
        <foaf:homepage rdf:resource="http://sunday.org/"/>
        <foaf:openid rdf:resource="http://sunday.org/"/>
        <foaf:img rdf:resource="http://gravatar.com/images/me.jpg"/>
    </foaf:Person>
</rdf:RDF>
```

The above XML snippet shows a basic FOAF vocabulary describing a person.

The triples below show the converted RDF N-Triples using EasyRDF converter[38].

```
<http://about.me#sunday> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://xmlns.com/foaf/0.1/Person> .

<http://about.me#sunday> <http://xmlns.com/foaf/0.1/name> "Sunday Ayandoku
n" .

<http://about.me#sunday> <http://xmlns.com/foaf/0.1/homepage> <http://sund
ay.org/> .

<http://about.me#sunday> <http://xmlns.com/foaf/0.1/openid> <http://sunday
.org/> .

<http://about.me#sunday> <http://xmlns.com/foaf/0.1/img> <http://gravatar.
com/images/me.jpg> .
```

[34] http://swoogle.umbc.edu/
[35] http://lov.okfn.org/
[36] http://www.dublincore.org/documents/dcmi-terms/
[37] http://xmlns.com/foaf/spec/
[38] http://www.easyrdf.org/converter

3.3.2 Semantically-Interlinked Online Communities (SIOC)

SIOC[39] focuses on the description of online community information, e.g. blogs, forums, mailing lists etc. Its main classes describe things like group, event, user, post, comment. SIOC has seen a noteworthy adoption via its usage in diverse software applications both commercial and open-source. [33] There are different SIOC exporters (e.g. WordPress SIOC Exporter, Drupal SIOC Exporter etc.) already implemented for a couple of popular weblogs, forums, and communities. Details about these exporters and other SIOC applications can be found here [34].

3.3.3 Good Relations

Good relations[40] is the web vocabulary for e-commerce. It describes e-commerce concepts ranging from products and business descriptions to pricing and method of delivery. It has a great impact in real-life applications. [32] It can be easily embedded into both static and dynamic web pages which are machine readable.

Other vocabularies include: Dublin Core[41], Simple Knowledge Organization System (SKOS[42]), Vocabulary of Interlinked Datasets (VoID[43]), VCard[44]

[39] http://www.sioc-project.org/
[40] http://www.heppnetz.de/projects/goodrelations/
[41] http://www.dublincore.org/documents/dcmi-terms/
[42] https://www.w3.org/TR/skos-reference/
[43] https://www.w3.org/TR/void/
[44] https://www.w3.org/TR/vcard-rdf/

3.4 Virtuoso

Virtuoso is a scalable cross-platform server that consolidates Relational[45], Graph[46,47], and Document Data Management with Web Application Server[48] and Web Services[49] Platform functionality, [35] providing data access, integration, and relational database management. [36] Due to its Linked Data deployment capabilities to provide a secure, high-performance, and cost-effective solution for exploiting the Linked Data Server capabilities; we have chosen it as our data store for storing user data in Graphs.

Virtuosos which is our linked data storage of choice implements the OAuth core 1.0 specification to grant access to specific user graph. Each user can generate a consumer key and secret on the Virtuoso's GUI for a specific virtuoso application to which the user is a member of e.g. the SPARQL[50] application. The token generated will be linked to the user account and the application instance. The following steps are the typical approach to establishing an authorized session to user graph using the consumer key and secret as described in details on Virtuosos OAuth implementation documentation[51].

1. Client request for access token via *request_token* to get a client id/secret pair to establish a session.
2. Client requests OAuth server for authorization using client id from step 1.
3. The client id from step 1 is used by the client to requests for an authentication token.
4. The authentication token from step 3 can be used to access data mapped with the client id from step 1.

In our own use case, the above implementation presents the following limitations.

- Each PDS user will need to access the virtuosos' GUI to generate consumer key and secret. This is an extra burden to a PDS user since we only want to present an easy to use PDS application to the user without burdening them with the underlying backend system.
- On the GUI, the user can only generate token for 2 applications. This is a big issue since users would like to grant access to many cloud applications that they interact with daily.

[45] https://en.wikipedia.org/wiki/Relational_model
[46] http://neo4j.com/developer/guide-data-modeling/
[47] https://en.wikipedia.org/wiki/Graph_database
[48] https://en.wikipedia.org/wiki/Application_server
[49] https://en.wikipedia.org/wiki/Web_service
[50] https://www.w3.org/TR/rdf-sparql-protocol/
[51] http://docs.openlinksw.com/virtuoso/voauth.html

Nevertheless, we are aware that virtuoso also has a full support implementation for our use case in the form of a fine-grained access control. But this feature is only available in the commercial[52] version of virtuoso since our proposed solution is an open-source project, we cannot go with this option.

3.5 Appscale

"*AppScale is an open source distributed software system that implements a cloud platform as a service (PaaS), enabling portable, scalable web application deployment.*" [4] It allows application developers run their apps that are built using the Google App Engine APIs on both public (such as Amazon EC2 and Google Cloud Engine) and private (such as OpenStack and Eucalyptus) cloud infrastructures.

What this means to application owners is that they can enjoy the same benefit they get when they use Google App Engine to power their application, even on Appscale. This is so because, Appscale is a complete clone of the GAE, that is an open source version of GAE.
AppScale users (developers) can take advantage of Google App Engine's quick improvement model while likewise guaranteeing that their applications remain portable. Furthermore, in terms of architecture design, AppScale could be described as a three-tier web Architecture with the following core components:

- **Application Servers**
- **Load Balancer**
- **Datastore**

These components are saddled with the responsibilities such as Deployment automation, Management, Scaling, and fault tolerance of the system and GAE applications.

[52] http://virtuoso.openlinksw.com/whats-new/

Alongside the above core components, AppScale supports [37] a list of APIs presented in the table below.

APIs	**Technology Used**
Datastore	AppDB[53]
Memcache	Memcached[54]
URL Fetch	urllib2[55]
Blobstore API	custom server built on Tornado[56]
XMPP[57]	ejabberd[58]
Channel API	ejabberd and strophejs[59]
Mail	sendmail
Images	Python Imaging Library (PIL)[60]
Task Queue	RabbitMQ[61]
Cron	Vixie Cron[62]
Search	SOLR[63]
CloudSQL	MySQL[64]
Users	AppScale Dashboard[65]
Routing, SSL[66]	Nginx[67]
Load balancing[68]	HAProxy[69]

Table 3: API supported within AppScale and how they are supported

[53] https://github.com/AppScale/appscale/wiki/AppDB
[54] http://memcached.org/
[55] http://docs.python.org/library/urllib2.html
[56] http://www.tornadoweb.org/
[57] https://en.wikipedia.org/wiki/XMPP
[58] https://www.ejabberd.im/
[59] http://strophe.im/strophejs/
[60] http://www.pythonware.com/products/pil/
[61] http://www.rabbitmq.com/
[62] https://wiki.gentoo.org/wiki/Cron
[63] http://lucene.apache.org/solr/
[64] https://www.mysql.com/
[65] https://github.com/AppScale/appscale/tree/master/AppDashboard
[66] http://info.ssl.com/article.aspx?id=10241
[67] https://www.nginx.com/
[68] https://f5.com/glossary/load-balancer
[69] http://www.haproxy.org/

Appscale automatically configures and deploy each of the required services. These services can be grouped into set of related services exposed via API

- Security & Authentication
- Monitoring & Logging
- Web Hosting
- User credentialing
- Messaging & Communication
- Data Storage & Processing

There are other components in AppScale, an overview which is presented in the figure below:

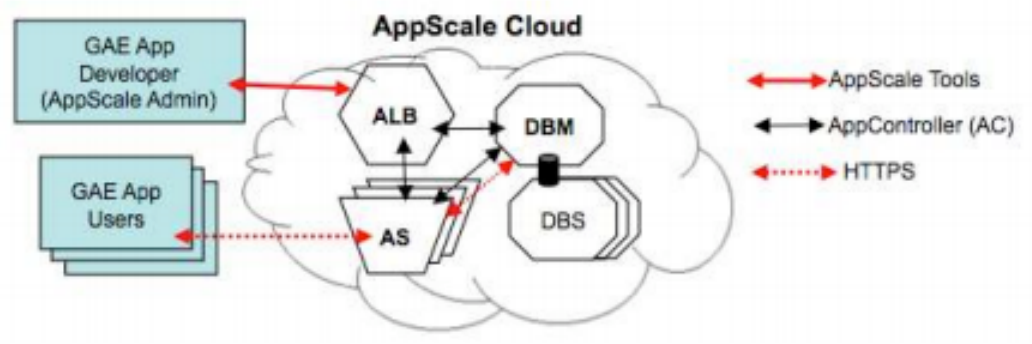

Figure 5: Overview of the AppScale design[70]

- **The AppScale tool** - a command line interface for interacting with the AppScale platform remotely. Such interactions include:
    - Deploy AppScale instance, Interaction with and administer AppScale instances and deployed GAE applications.

- **AppServers (AS)** – these are engines that aids the interactions between GAE applications and Database Master (DBM) for data storage and access. The interactions are achieved via HTTPS. Application users also interact with the **AS.**
- **Database Management System** – the system that facilitates distributed scalable, and fault tolerance data management.

[70] https://www.cs.ucsb.edu/~ckrintz/papers/cloudcomp09.pdf

- **AppController (AC)** – The component that enables inter-component communication. It is also responsible for setup, bootstrapping, and shutting down of AppScale instances. Other responsibilities include:
    - Deployment and Authentication of deployed applications
- **AppLoadBalancer (ALB)** – functions as the deployment head node and establishes a connection to GAE applications running in **AppServers.**

For every AppScale deployment, there is only one **ALB** which is considered the head node, at least one AppServers, one DBM and one or more DBSs. A single node can implement any of the individual components or a combination of components. To ensure a secure interaction among systems, communications are encrypted via the secure socket layer (SSL). [38]
We have chosen AppScale as our application PaaS for the PDS application because it offers a scalable cloud platform that integrates, extends existing web service, and empowers users to deploy cloud technologies easily on premise or on their preferred public cloud. [38]

3.6 Django Web Framework

Django[72] is a high-level Python Web framework, developed to make repeated Web-development tasks swift and with less difficulty. We have chosen Django as our framework for this project because it offers a quick development and clean, realistic design approach. Also, our proposed solution requires a conscious security concern, Django on its own takes security as a priority. Another area of consideration is scalability and Django's ability to rapidly and resiliently scale is a big plus for us to consider it. More so, AppScale as our platform of choice has the majority of its codebase written in python. Hence, we believe working with Django will help us understand better the underlying design of the AppScale platform.

[72] https://www.djangoproject.com/

3.7 OAuth 2.0

OAuth 2.0 is an authorization framework which is an HTTPS-based protocol that empowers application end-users (Resource owner) to grant third-party application (Client/Consumer) limited access to secured resources on the server. There are four major roles in OAuth which includes:

- Resource Owner
- Client
- Resource Server
- Authorization Server

Figure *6*: OAuth 2.0 authorization flow

Figure 6 shows the interaction among these roles.

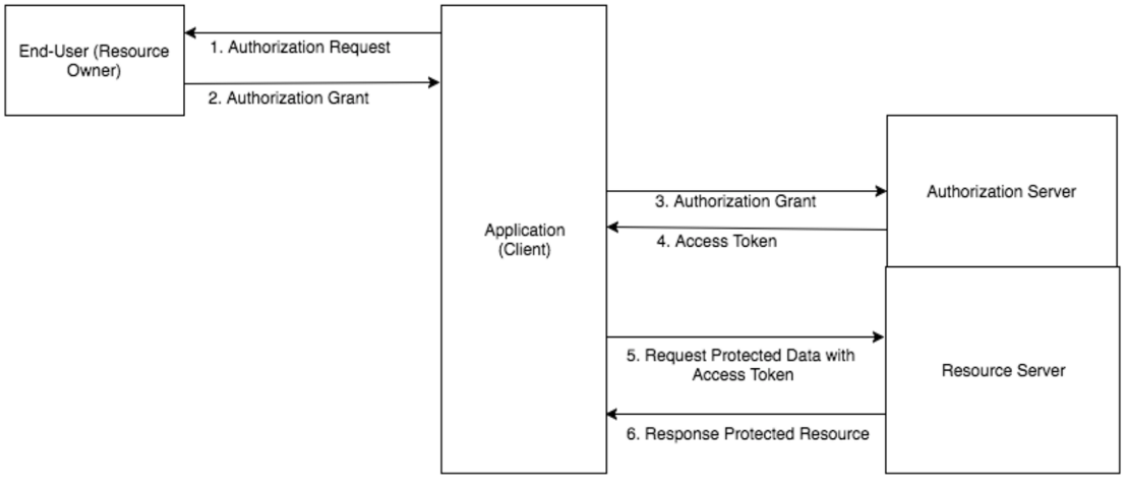

Figure 6: OAuth 2.0 authorization flow

In Figure 6 we can see the general idea of OAuth authorization flow. Some of these details would be described in chapter 4 of the actual implementation for our use case.

1. The client asks an authorization request from the end user. If granted, the client gets an authorization code.
2. The authorization grant is used to request for an access token from the Authorization server.
3. The Authorization server authenticates the client and checks the grant validity before issuing the access token.
4. The clients then use the access token to request for the protected resource.

5. The required resource is sent back by the resource server if the access token is valid. A detailed description of OAuth 2.0 can be found here [39].

3.8 Tyk

Tyk is an open source API Gateway, that allows API owners control who accesses their API, when and how they access it. Tyk features include an API gateway, analytics, developer portal and dashboard. [40]

In [41], we have seen detailed key features of Tyk. These features present to us with what we need to implement a secure, easy to use and flexible control on user personal data. These features include:

- **RESTFul API** – This feature makes it very interesting that, everything that can be done on the Tyk GUI can be achieved programmatically from our own system. And since most of our interaction with Tyk will be done from our system, the available set of APIs make things easier.
- **Multiple access protocols** – Tyk supports multiple authentication protocols which include OAuth 2.0, Standard access tokens[73], HMAC[74] Signatures, Basic Authentication[75], JWT[76] and Keyless access[77] methods.
- **Quotas** – Tyk allows API owners to enforce usage quotas on a per-key basis.
- **Granular Access Control** – Tyk can grant access to an API in a granular form. i.e. A key can access for example only contact details of a particular user.
- **Key Expiry** – when creating keys, you can explicitly tell when the key will expire.
- **API Versioning** – It offers flexible API Versioning.
- **Blacklist/Whitelist/Ignored endpoint access**
- **Analytics logging** – It can log detailed usage data on who is accessing the API's.
- **Webhooks** – It can Trigger webhooks against events e.g. access token generation.
- **IP Whitelisting**
- **Zero downtime restarts** – the service can restart after applying changes in the configuration without affecting any active request.

[73] https://tyk.io/docs/tyk-api-gateway-v-2-0/access-control/standard-access-tokens/
[74] https://tools.ietf.org/html/rfc2104
[75] https://en.wikipedia.org/wiki/Basic_access_authentication
[76] https://jwt.io/
[77] https://tyk.io/docs/tyk-api-gateway-v-2-0/access-control/keyless-access/

4. System Design & Architecture

The proposed solution is a personal data storage hereby referred to as PDS. It provides a self -contained, application agnostic personal data repository for individual users, which is in total control of the user. The data are stored in standard linked data format encoded RDF triples. It adopts a decentralized architecture which opposes the most widely used centralized architecture by various web applications. In the centralized approach, the service providers as the custodian of enormous user data, have unprecedented amounts of data about the behavior and personalities of individual [42] . As user privacy concern grows, we believe a solution like this would offer cloud users more trust as they interact with cloud applications.

As PDS user interacts with other cloud applications that require their data, the user can delegate the external application to fetch data on their behalf, by granting access and authorization until such access is revoked.
The authorization and access revoke will be implemented using OAuth 2.0 protocol a similar approach discussed earlier in the case of OpenPDS[78].

[78] github.com/HumanDynamics/openPDS/blob/master/doc/openPDS%20Developer%20Documentation.pdf

4.1 Personal Storage And Related Mechanisms

In this section, we describe the architecture of the proposed solution and other related mechanisms of the implementation.

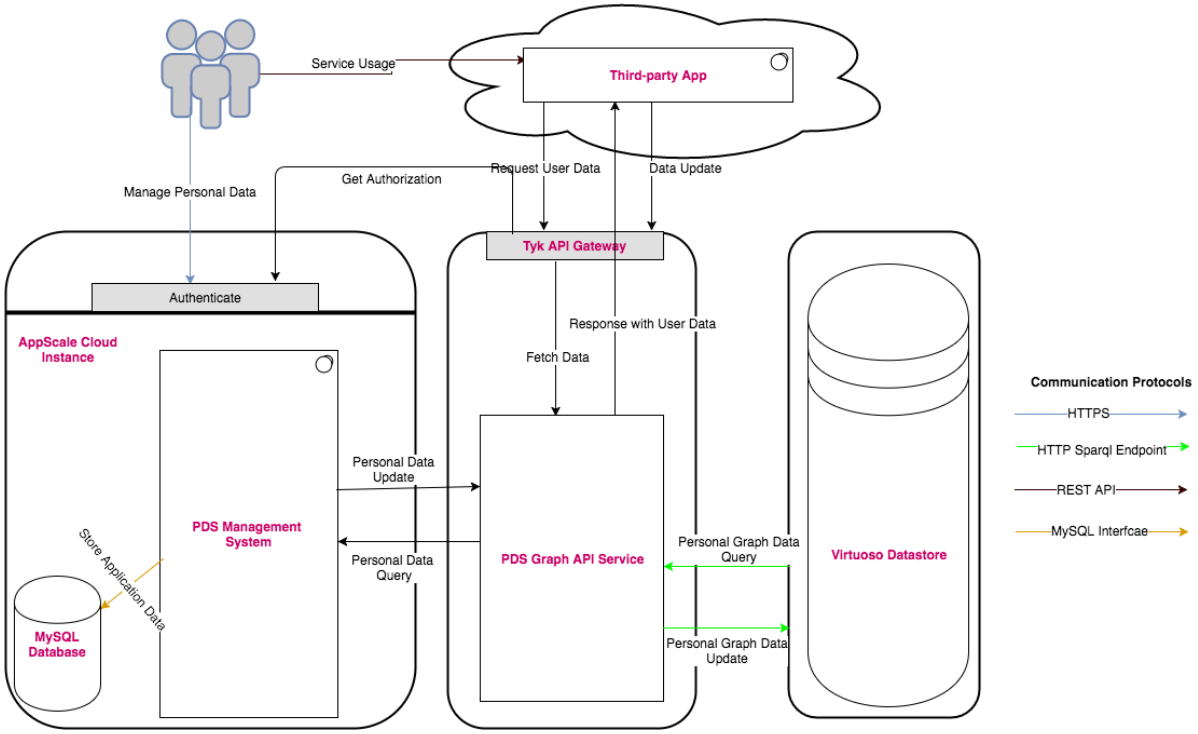

Figure 7: PDS system architecture

Figure 7 is an overview of the PDS design showing how different components interact with each and the protocol for such interactions.

4.1.1 PDS Management System

This is the PDS application running on the AppScale platform. It was developed using the Django Web framework. The user interacts with their data store using this app. Its front-end is built with Angularjs[79] which interact with the back-end via a REST API.

It has a user dashboard - to manage personal data (create, update, etc.), manage connected apps, grant and revoke third-party app access.

It enables users to see various graphs that represent a different section of their data e.g. personal graph, emails, addresses, telephones, online accounts etc. Whenever a user update or add of personal information, the PDS Application encodes the data in RDF triples and send it to the PDS service via HTTP RESTful API for further processing.

Sample encoded RDF triples for the email graph

[79] https://angularjs.org/

Each quoted set of URIs is a triple containing ***<subject><predicate><object>***:

"<https://graph.ir.ee/users/1/persons> <http://www.w3.org/2006/vcard/ns-hasEmail> <https://graph.ir.ee/users/1/emails/sunday-ayandokun-ut-ee> ."

"<https://graph.ir.ee/users/1/emails/sunday-ayandokun-ut-ee> <http://www.w3.org/2006/vcard/ns-hasValue> <mailto:sunday.ayandokun@ut.ee> ."

"<https://graph.ir.ee/users/1/emails/sunday-ayandokun-ut-ee> <http://www.w3.org/1999/02/22-rdf-syntax-ns -type> <http://www.w3.org/2006/vcard/ns-Work> ."

The above triples use the VCard[80] ontologies.

4.1.2 PDS Graph API Service

This is the main User Data Graph API service that responds to every request to store and retrieve user data. Whenever the service receives data insertion or update requests, it communicates with Virtuoso server via a HTTP SPARQL endpoint to store the triples in their respective graphs. If the request is to get user data, it must pass through the Tyk API gateway for user authorization. As a proof of concept implementation, the PDS service currently, only add and update user data via PDS application. In the future, data update from external services will be implemented.

[80] https://www.w3.org/TR/vcard-rdf/

4.1.3 Virtuoso Datastore

This is the datastore that house individual user graph. When a user setup their PDS, a user account is created on the Virtuoso server for the user. This account is used to create the user graphs with predefined permissions on those graphs. These permissions are granted to only that user on those graphs.

The steps to set these permissions are given below, for a sample user:

1. Make sure no user on the system has permission on any graph
   DB.DBA.RDF_DEFAULT_USER_PERMS_SET ('nobody', 0);
2. Create user - **DB.DBA.USER_CREATE ('username', 'password');**
3. Grant Sparql update to the user - **GRANT SPARQL_UPDATE TO "username";**
4. Set permission to none - **DB.DBA.RDF_DEFAULT_USER_PERMS_SET ('username', 0);**
5. Create user graph - **CREATE GRAPH <https://graph.ir.ee/users/1/emails>**
6. Set read access for the user on the created graph
   DB.DBA.RDF_GRAPH_USER_PERMS_SET('https://graph.ir.ee/users/username/emails', 'username',1);
7. Set write access for the user
   DB.DBA.RDF_GRAPH_USER_PERMS_SET('https://graph.ir.ee/users/username/emails','username', 3);

The following are the set of graphs identified, as related to inforegister.ee users' data, which is our users base for validating this concept.

- https://graph.ir.ee/users/<user_id>/persons
- https://graph.ir.ee/users/<user_id>/accounts
- https://graph.ir.ee/users/<user_id>/telephones
- https://graph.ir.ee/users/<user_id>/emails
- https://graph.ir.ee/users/<user_id>/addresses
- https://graph.ir.ee/users/<user_id>/facebook
- https://graph.ir.ee/users/<user_id>/twitter
- https://graph.ir.ee/users/<user_id>/linkedln
- https://graph.ir.ee/users/<user_id>/public-contracts
- https://graph.ir.ee/users/<user_id>/preferences
- https://graph.ir.ee/users/<user_id>/monitoring-organizations

Sample SPARQL query and result

SPARQL SELECT * WHERE {GRAPH <https://graph.ir.ee/users/1/emails> {?s ?p ?o }};

Query result:

s *ANY*	p *ANY*	o *ANY*
https://graph.ir.ee/users/1/emails/sunday-ayandokun-ut-ee	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2006/vcard/ns#Work
https://graph.ir.ee/users/1/emails/sunday-ayandokun-ut-ee	http://www.w3.org/2006/vcard/ns#hasValue	mailto:sunday.ayandokun@ut.ee
https://graph.ir.ee/users/1/persons	http://www.w3.org/2006/vcard/ns#hasEmail	https://graph.ir.ee/users/1/emails/sunday-ayandokun-ut-ee

No. of rows in result: 3

Figure 8: Sample SPARQL query result on the user emails graph

Query result:

s *ANY*	p *ANY*	o *ANY*
https://graph.ir.ee/users/1/addresses/akadeemia-tee-418v	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2006/vcard/ns#Home
https://graph.ir.ee/users/1/persons	http://www.w3.org/2006/vcard/ns#hasAddress	https://graph.ir.ee/users/1/addresses/akadeemia-tee-418v
https://graph.ir.ee/users/1/addresses/akadeemia-tee-418v	http://www.w3.org/2006/vcard/ns#street-address	Akadeemia Tee,418V
https://graph.ir.ee/users/1/addresses/akadeemia-tee-418v	http://www.w3.org/2006/vcard/ns#locality	Tallinn
https://graph.ir.ee/users/1/addresses/akadeemia-tee-418v	http://www.w3.org/2006/vcard/ns#postal-code	122222
https://graph.ir.ee/users/1/addresses/akadeemia-tee-418v	http://www.w3.org/2006/vcard/ns#country-name	Estonia

No. of rows in result: 6

Figure 9: Sample SPARQL query result on the user addresses graph

Query result:

s *ANY*	p *ANY*	o *ANY*
https://graph.ir.ee/users/1/telephones/00372000000000	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2006/vcard/ns#Mobile
https://graph.ir.ee/users/1/persons	http://www.w3.org/2006/vcard/ns#hasTelephone	https://graph.ir.ee/users/1/telephones/00372000000000
https://graph.ir.ee/users/1/telephones/00372000000000	http://www.w3.org/2006/vcard/ns#hasValue	tel:+372000000000

No. of rows in result: 3

Figure 10: Sample SPARQL query result on the user telephones graph

Query result:

s *ANY*	p *ANY*	o *ANY*
http://www.xbox.com/live/sunnepah	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://xmlns.com/foaf/0.1/OnlineAccount
http://www.xbox.com/live/sunnepah	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://xmlns.com/foaf/0.1/OnlineGamingAccount
http://www.xbox.com/live/sunnepah	http://xmlns.com/foaf/0.1/accountServiceHomepage	http://www.xbox.com/live/
http://www.xbox.com/live/sunnepah	http://xmlns.com/foaf/0.1/accountName	sunday.ayandokun@live.com
https://graph.ir.ee/users/1/accounts/sunday-ayandokun	http://xmlns.com/foaf/0.1/account	http://www.xbox.com/live/sunnepah

No. of rows in result: 5

Figure 11: Sample SPARQL query result on the user public-contracts graph

Query result:

s *ANY*	p *ANY*	o *ANY*
https://graph.ir.ee/users/1/persons/sunday-ayandokun	http://graph.ir.ee/hasContract	https://graph.ir.ee/users/1/public-contracts/123435353-24253253532
https://graph.ir.ee/users/1/public-contracts/123435353-24253253532	http://purl.org/dc/terms/title	Inforegister Contract
https://graph.ir.ee/users/1/public-contracts/123435353-24253253532	http://purl.org/dc/terms/description	Decription
https://graph.ir.ee/users/1/public-contracts/123435353-24253253532	http://purl.org/procurement/public-contracts#durationDays	365
https://graph.ir.ee/users/1/public-contracts/123435353-24253253532	http://purl.org/procurement/public-contracts#startDate	2015-12-04
https://graph.ir.ee/users/1/public-contracts/123435353-24253253532	http://purl.org/procurement/public-contracts#actualEndDate	2016-12-04
https://graph.ir.ee/users/1/public-contracts/123435353-24253253532	http://purl.org/procurement/public-contracts#cancellationDate	2015-12-04
https://graph.ir.ee/users/1/public-contracts/123435353-24253253532	http://graph.ir.ee/public-contracts/comfirmation-type	Cancelled
https://graph.ir.ee/users/1/public-contracts/123435353-24253253532	http://purl.org/procurement/public-contracts#contractPrice	45.99
https://graph.ir.ee/users/1/public-contracts/123435353-24253253532	http://purl.org/procurement/public-contracts#attachment	https://graph.ir.ee/users/1/public-contracts/document/20140929144237_
https://graph.ir.ee/users/1/public-contracts/123435353-24253253532	http://purl.org/dc/terms/creator	https://graph.ir.ee/users/1/persons/sunday-ayandokun

No. of rows in result: 11

Figure 12: Sample SPARQL query result on the user accounts graph

4.2 PDS User Access Control

In this section, we describe in details the user access control mechanism for granting third-party access to private data. In our attempt to implement a user-centric authentication mechanism, one that is easy to use and allows users to share their data securely; we have thought of various options. The idea is to follow the popular oauth2 three-legged authorization flow described in the previous section. When a PDS user is using a third-party application – a consumer in the oauth2 context, that requires the user data, the user is redirected to their PDS application to grant access if the user is not logged-in to PDS, they are required to logged-in with their Appscale account credentials. Once logged-in, the user is prompted to grant access to the requesting application. If granted the user has just authorized the external application to access their private data until the access is revoked. We discuss next the various implementation options.

4.2.1 Tyk OAuth 2.0 And Django OAuth Toolkit Access Control

In our search for another alternative to virtuoso OAuth for controlling access to user data, we considered first the Django OAuth Toolkit[81] – a Django pluggable app that can serve as an OAuth provider. It makes use of the popular python OAuthLib[82] to ensure a RFC-compliant[83] and provides an end-to-end OAuth2 authorization flow. As good as this seems to serve our use case, it lacks other key features that we would like to provide e.g. API management dashboard, Developer portal, Usage metrics etc. And this is where Tyk comes in.

As we described earlier, Tyk is an open-source API management platform capable of providing OAuth2 authorization, with a well-defined set of APIs to interact with the Tyk API gateway.

Table 4 describes the related endpoints.

[81] https://django-oauth-toolkit.readthedocs.org/en/latest/index.html
[82] http://oauthlib.readthedocs.org/en/latest/installation.html
[83] http://tools.ietf.org/html/rfc6749

#	**Endpoints**	**Descriptions**
1	/clients/create	It handles client (consumer) creation
2	/oauth/authorize/	It handles clients authorization request, which prompts resource owner authorization
3	/tyk/oauth/authorize-client/	Handles Tyk clients authorization and respond with an authorization code
4	/oauth/token	Handles client request for access token using the authorization code granted earlier.

Table 4: Tyk OAuth 2.0 related endpoints

In order to have a full integration between Tyk and the PDS application, we need a mechanism for binding client-token pair generated by Tyk to our users' identity. With this in place, we know which user has granted access to which external application, so that each user can view the activities of those applications on their dashboard and can as well revoke the access when the need arises.

4.2.2 PDS OAuth 2.0 Authorization Flow

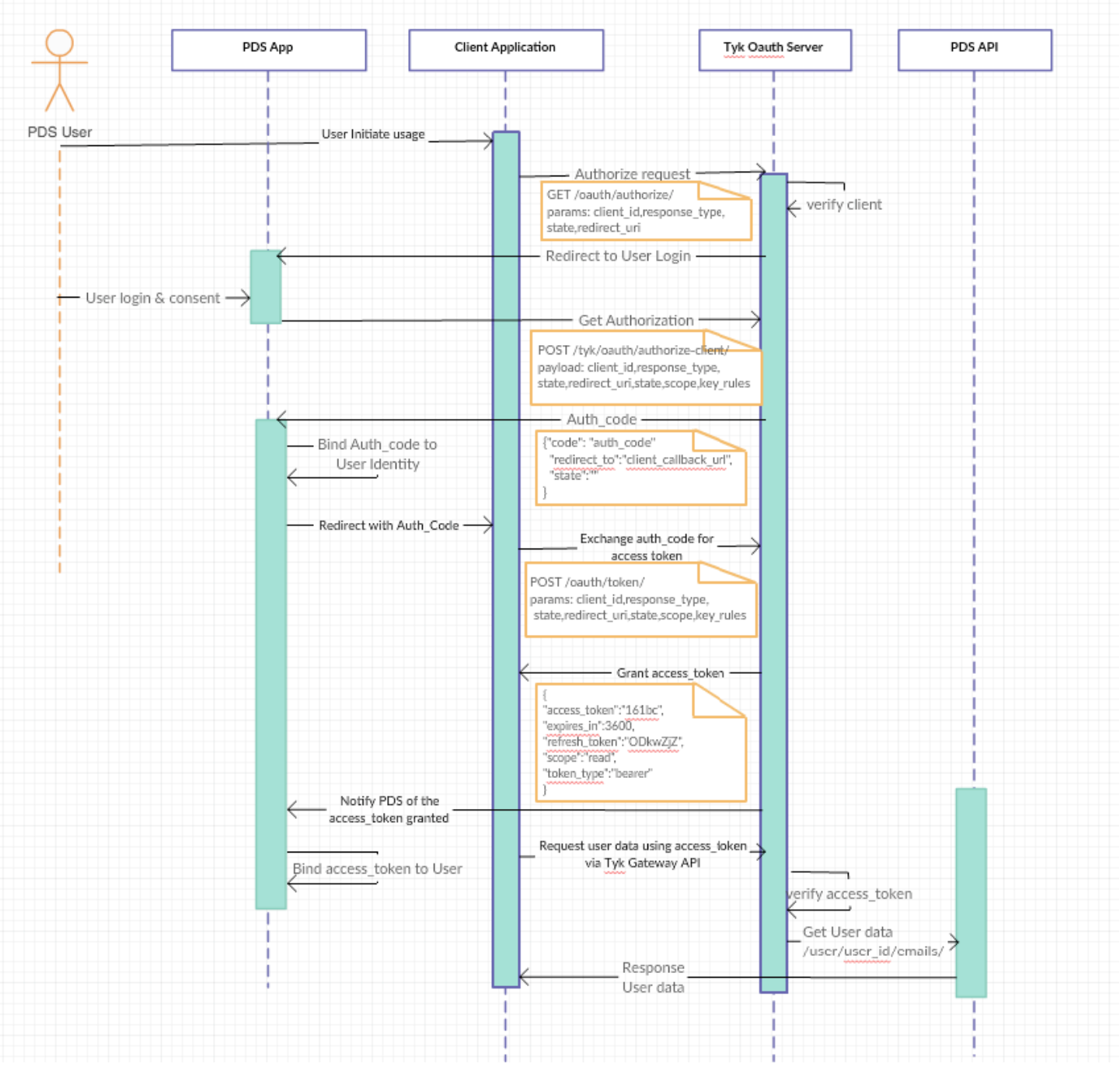

Figure 13: Sequence diagram for the PDS OAuth 2.0 authorization flow

sequence diagram shows a detailed flow of the authorization process in our implementation, which is not any different from how others have implemented OAuth 2.0 authorization in their services.

In '**Get Authorization**' step, the **key_rules** parameter gives us the ability to achieve a fine-grained access control with Tyk.

Figure 14: Sample access_rights in the key_rules

Figure 14 shows **access_rights** in **key_rules** containing **allowed_urls** parameter which tells Tyk gateway that the token generated with this **key_rules** should be allowed to access only the given endpoints when a **GET** request is made. Otherwise, access is denied with error message:

```
{
    "error": "Access to this resource has been disallowed"
}
```

It ensures that users have a granular access control to their data. This is the mechanism we used to achieve one of the 7 privacy by design principles ***'Privacy as the Default Setting'*** [43]. This ensures that user data are automatically protected since only data needed by a specific application is granted access to, which is a kind of endpoints filtering based on requesting application privileges. A similar approach in [44], where the researchers embedded privacy into the use of a personal data vault on mobile devices and provides a set of filters controls that minimizes the movement of data from the vault to third party applications. According to [12] *"This fulfils Privacy requirement of data minimization at every stage of the information lifecycle, and if personal information is not needed, it should never be collected in the first place"*.

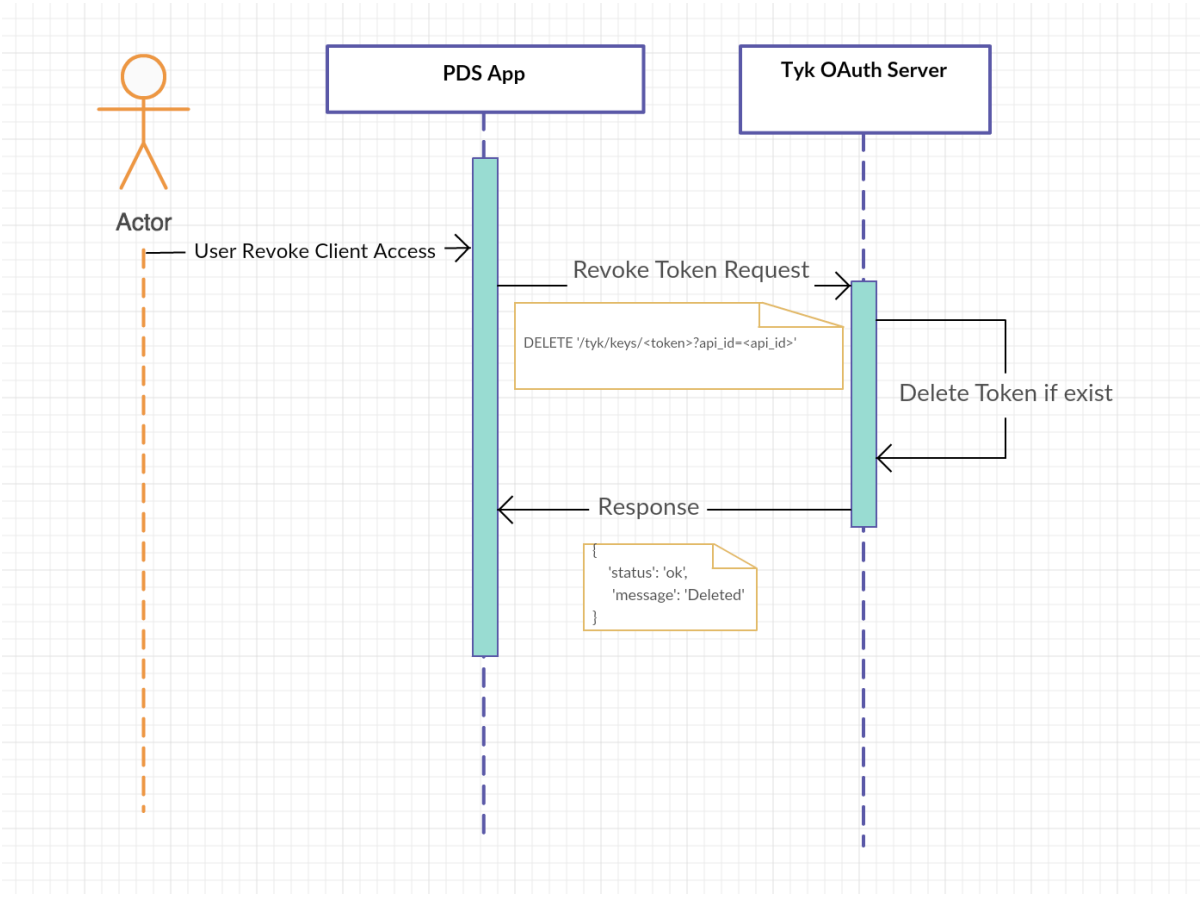

Figure 15: Token revoke sequence diagram

shows a flow of how a user can revoke access from a client that was granted access to earlier.

4.3 Graph API Documentation

"A Graph API is a RESTful, user-centric, hypermedia API that organizes web resources under a unified meta-model of Objects, Aggregations of objects and connections towards them which are created by users. It is based on a common dictionary and it includes a minimum set of properties in order to reduce time and cost of connection and integration with other APIs." [22]

This definition is the most comprehensive definition of the Graph API that we have come across. The PDS Graph API documentation with detailed sample requests and responses can be found in the Appendix I.

5. Proof Of Concept Implementation With Inforegister.ee

In order to validate this implementation, we have migrated **inforegister.ee** user data to the personal data storage. The aim of this task was to make personal user data at inforegister.ee, which has been collected and is actively maintained via various sources, such as Facebook, LinkedIn, Twitter etc, available to other applications such that they only need to implement a single interface for consumption of personal data. The task represents a case study, which demonstrates the challenges faced while migrating the users from the legacy application to the AppScale platform which has an identity management component.

Migration to AppScale platform presents the following scenarios:

Scenario 1: No previous users to be migrated - This is a situation whereby there is no previous users to be migrated to AppScale platform, users will create fresh accounts.

Scenario 2: There are users to be migrated – In this case, there are existing users on the previous platform which needs to be migrated to AppScale platform.

Inforegister.ee falls into the second scenario. Hence, there is a need to create an account for each user on AppScale from the previous user base, on behalf of the users. The migration task includes the following plans:

- Import users from CSV file containing users email using the batch user import command in AppScale-tool[84].
- The tool creates account for each user and generates random password
- Users will receive email to reset password on AppScale

Although, AppScale has a datastore which can be used to store all user data and application data from the previous **inforegister.ee** platform. But, the implication of this is, redesigning the entire Inforegister schema to fit into AppScale Datastore. We see that this is not a cost effective approach for a proof of concept implementation, with **inforegister.ee** schema having over 300 tables.

With that in mind, there is a need to create a reference table to map each newly created AppScale user to their previous data.

[84] https://github.com/AppScale/appscale-tools

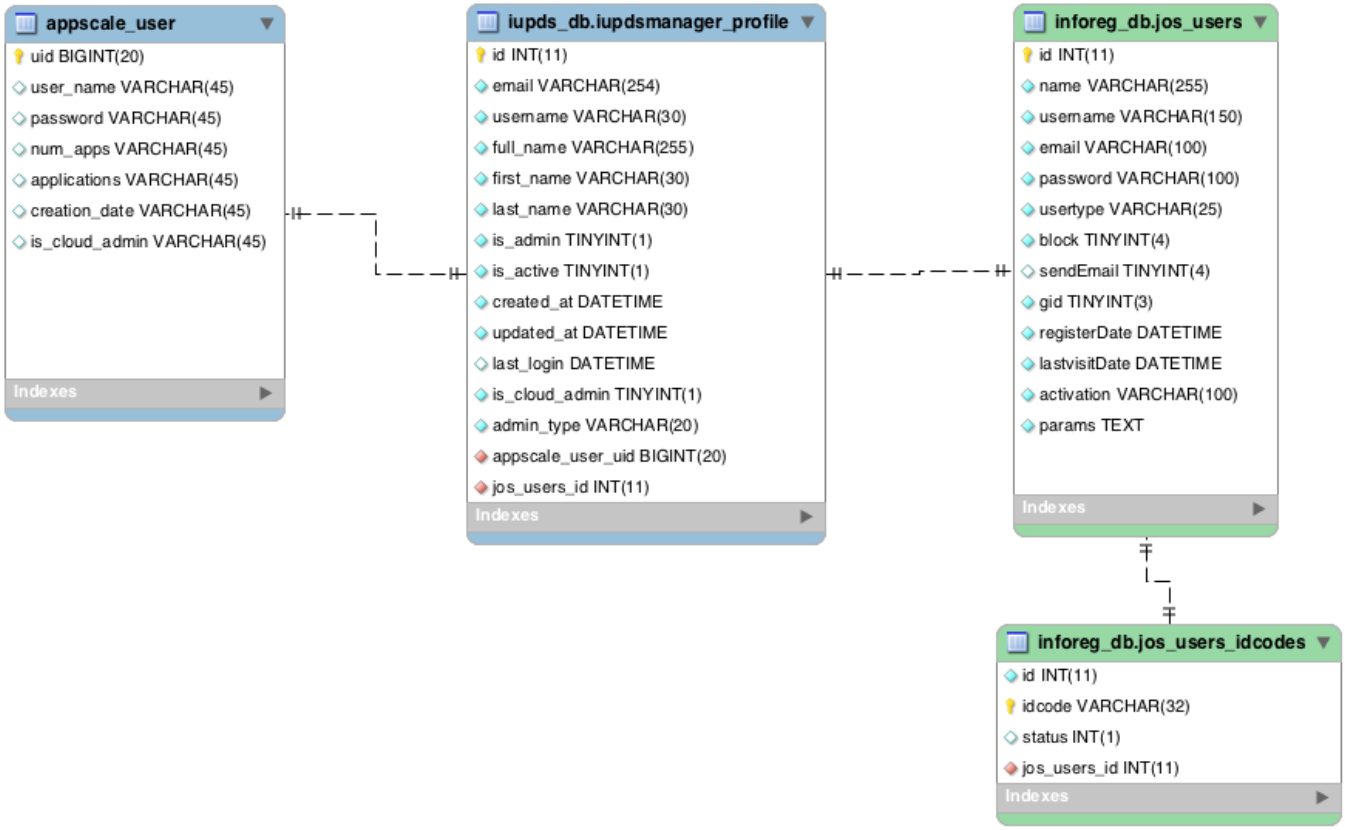

Figure 16: ER-diagram showing mapping of AppScale users with their data on inforegister.ee schema

In ER-diagram, the table **appscale_user** shows the AppScale user entity. Table **inforeg_db.jos_users** is the user entity for the current user platform with another table showing the relationship. There are much more tables in the current Inforegister schema that has relationship with the **inforeg_db.jos_users** table. This is the reason for the table in the middle (**iupds_db.iupdsmanager_profile**) which links the **appscale_user** to the rest of their data. The relationship table is application specific, that is any application that wants to migrate to AppScale platform must find a way to manage this relationship.

NOTE: The **batch user import command** is not currently available in AppScale-tools. This is a functionality we have extended in AppScale and we intend to propose this to the AppScale team as part of our contribution to the AppScale Open-Source.

6. Performance Evaluation

We carried out a couple of tests to see how the system will perform under different workloads and the results are shown below.

6.1 Tyk's Average Request Response Time As Tokens and Requests Increase

First, we measured how Tyk API gateway performs under load with respect to a number of keys in the system. In these use case with full OAuth 2.0 flow, the gateway performs the following during each request:

- It validates the client making the request
- It validates the access token in the incoming request against access control
- It checks if the token has not exceeded the quota
- It checks if the request is within limit rate (e.g. 1000 request/sec)
- It records analytics
- If the above validations are fine, it then proxy the request to the API and returns the response to the client

The Tyk Server running on a local virtual machine has the following specifications:

- 2 CPUs
- 2 GB RAM
- Ubuntu 64 bit

The table below shows different scenarios as the number of keys increases, as hit rates rise and the average response time for each case.

Number of Tokens	Total Request(10 sec)	Average Response Time
100000	500	0.05477332
200000	1000	0.265296508
300000	1500	0.265692879
400000	2000	1.012392513
500000	2500	0.513143449
600000	3000	1.351999548
700000	3500	1.384723209
1000000	4000	1.369384492

Table 5: Tyk performance measure test data

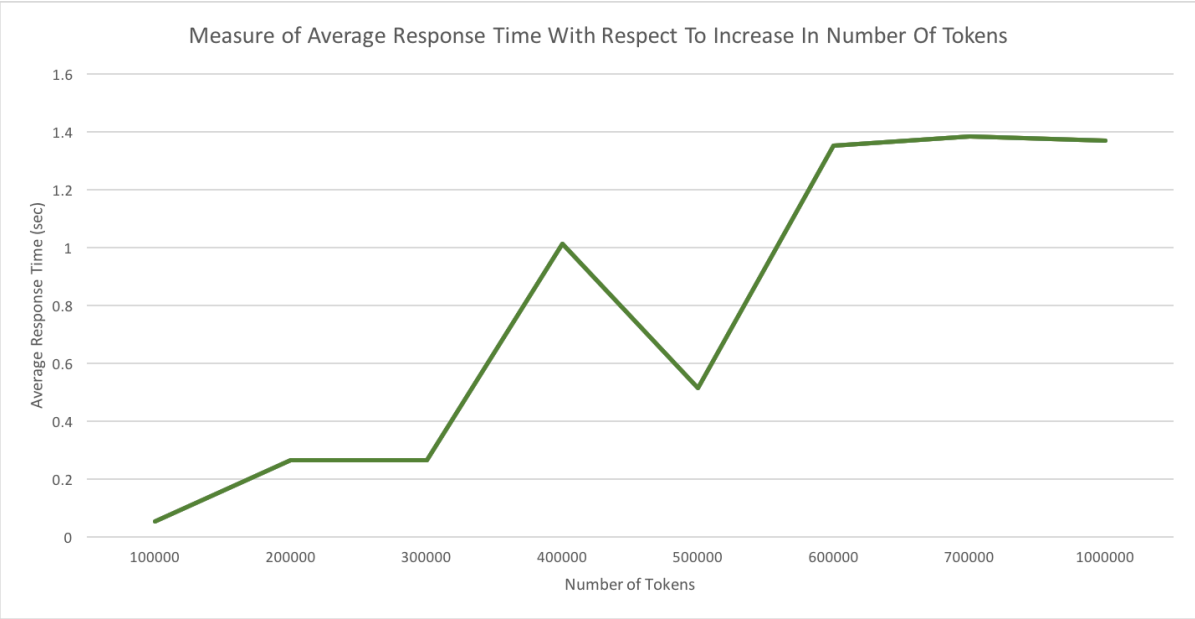

Figure 17: Chart showing Tyk average request response time as number of token increases

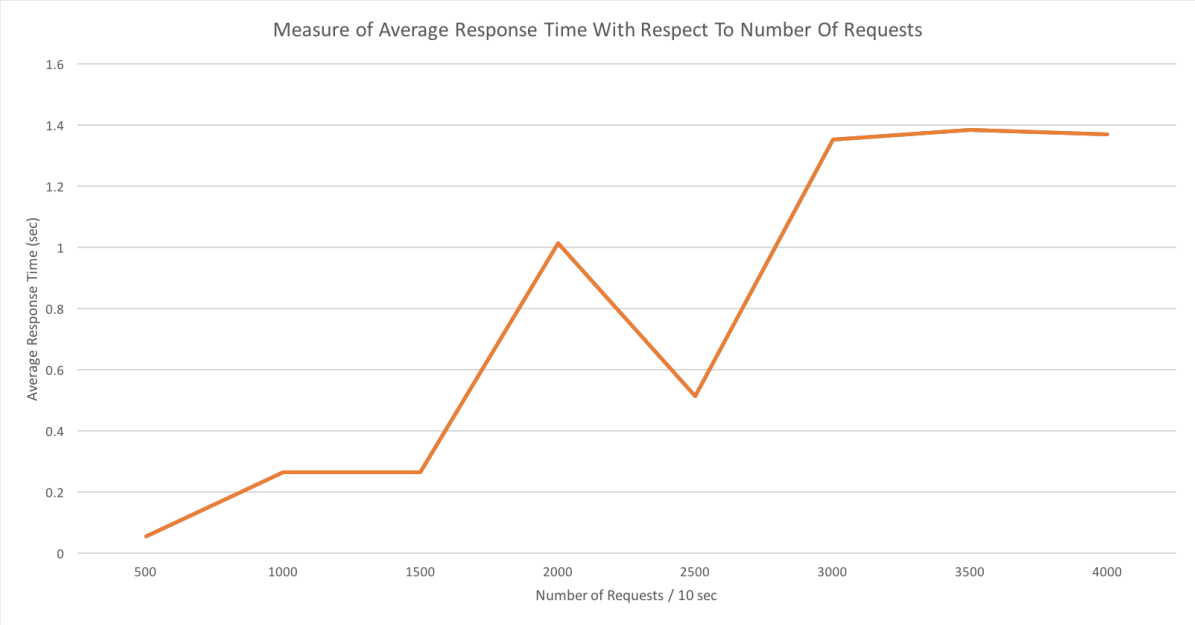

Figure 18: Chart showing Tyk average response time as number of request increases

We wanted to see how request response time is impacted when the number of tokens in the system increases and also, when the number of requests within a 10 seconds frame increases. As we can see above, the average response time increases as the number of tokens increases and also, when the number of request increases. Interestingly, from about 3000 requests and above the response time becomes steady. Since our main focus is not to test the overall Tyk performance, a detailed performance test by Tyk's team can be found here [45].

6.2 AppScale Server CPU Usage with Response Time During User Data Import

Here we want to compare how data import response time is impacted by system CPU usage, during import of existing user data into AppScale platform. This measure is important in order to understand resource utilization during data import and be readily prepare when doing such.

Number of Requests	Average Response Time	CPU Usage %
1000	0.046546682	43
2000	0.046997616	73.25
3000	0.051837256	83.95
4000	0.058330669	104
5000	0.05808148	112.7
6000	0.063637054	112.7
7000	0.063122225	80.6
8000	0.066253425	80.6

Table 6: AppScale user import response time with CPU usage results

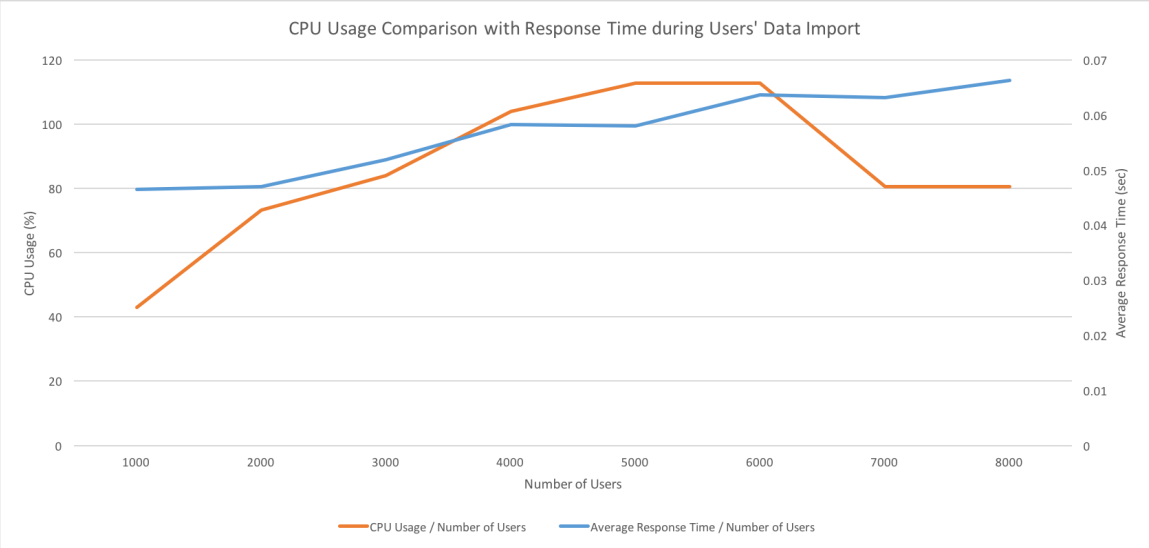

Figure 19: Chart showing CPU usage with response time during user data import

In the chart above, we see an increase in response time as user import rate increases, with CPU usage also going up. It is also worth mentioning that during the import, AppScale dashboard application takes a longer time to load. Hence, data import can impact user accessibility to the system. And such operation is better done during the off-peak period.

In addition, this test was carried out with AppScale's default deployment configuration, i.e. all components running on a single node. Better performance can be achieved by using a more advanced deployment configuration to benefit from AppScale's auto-scale capability.

7. Conclusion & Future work

The thesis describes design and implementation of an application-agnostic personal storage. The storage was implemented as a Graph API to allow sharing user data among applications and is using the linked data principles to ensure data usability throughout various use cases.

In addition, as part of a proof of concept implementation we have enhanced AppScale platform with a Personal Data System (PDS) component, with data management and dashboard functionalities. The PDS component allows users to manage their data and control access at the granularity level of data classes. This fine-grained access control was achieved by integrating the PDS and its Graph API with the Tyk API Gateway in order to have a full OAuth 2.0 authorization support. The current implementation allows users to store different classes of their personal data at the PDS. Users can also grant access to their data to third-party applications and revoke such access at any time.

In the future, we would like to extend PDS functionalities to allow data update from external services. For example, if a user wants to allow an application to store its bank transactions, social media data or health related data on the PDS, this will be possible via an APIs as well. Finally, as part of our effort to incorporate privacy by design in our implementation, we will look more deeply into privacy issues surrounding personal data storage services.

The implementation source code is available on Github at the following addresses:
PDS Management System - https://github.com/Sunnepah/iupds-appscale
PDS Graph API - https://github.com/Sunnepah/pdsservice
PDS Client - https://github.com/Sunnepah/pds-client-app

8. References

[1] Q. Hassan, "Demystifying Cloud Computing," *The Journal of Defense Software Engineering,* p. 16–21., Jan/Feb 2011.

[2] P. Mell and T. Grance, "The NIST Definition of Cloud Computing," 2011.

[3] E. Olden, "Architecting a Cloud-Scale Identity Fabric," *Computer,* vol. 44, no. No.03, pp. 52-59, 2011.

[4] (. Chandra Krintz, "The AppScale Cloud Platform - Enabling Portable, Scalable Web Application Deployment.," 2013. [Online]. Available: http://www.cs.ucsb.edu/~ckrintz/papers/IC2013.pdf . [Accessed 2015].

[5] D. N. T. W. C. L. M. W. C. (. dos Santos, "Privacy-preserving Identity Federations in the Cloud - A Proof of Concept," *Int. J. Security and Networks,* vol. 9, no. 1, 2014.

[6] G. (. Goth, "Privacy gets a new round of prominence," *IEEE Internet Computing,* vol. 15, pp. 13 - 15, 2011.

[7] C. M. a. I. W. W. Kuan Hon, "The problem of 'personal data' in cloud computing: what information is regulated? - The cloud of unknowing," *International Data Privacy Law,* vol. 1, no. 4, pp. 211-228, 1 4 2011.

[8] D. R. (. 2. Ann Cavoukian, "Big Privacy: Bridging Big Data and the Personal Data Ecosystem through Privacy by Design," December 2013. [Online]. Available: https://www.ipc.on.ca/site_documents/PbDBook-From-Rhetoric-to-Reality-ch3.pdf. [Accessed 21 12 2015].

[9] B. A Digital Single Market for Europe: Commission sets out 16 initiatives to make it happen, "European Commission," 16 May 2015. [Online]. Available: http://europa.eu/rapid/press-release_IP-15-4919_en.htm.

[10] A. Systems, "Appscale," [Online]. Available: https://github.com/AppScale/appscale. [Accessed 12 10 2015].

[11] "Wikipedia," [Online]. Available: https://en.wikipedia.org/wiki/Personal_Data_Service. [Accessed 24 11 2015].

[12] P. Ann Cavoukian, *Privacy by Design and the Emerging Personal Data Ecosystem,* Ontario, 2012.

[13] Personal.com, "Personal Docs," [Online]. Available: http://docs.personal.com/index.html. [Accessed 27 12 2015].

[14] IDHOLE, "IDHOLE," [Online]. Available: https://www.idhole.com/. [Accessed 11 12 2015].

[15] LockerProject, [Online]. Available: http://lockerproject.org/. [Accessed 15 12 2015].

[16] Singly, "Singly," [Online]. Available: https://github.com/Singly/hallway. [Accessed 03 05 2016].

[17] S. S. W. A. (. P. Yves-Alexandre de Montjoye, "On the Trusted Use of Large-Scale Personal".

[18] Media Lab, Massachusetts Institute of Technology, Cambridge, Massachusetts, United States of America , "OpenPDS Personal Data with Privacy," [Online]. Available: http://openpds.media.mit.edu. [Accessed 23 12 2015].

[19] IETF OAuth WG, "OAuth 2.0 protocol," [Online]. Available: http://oauth.net/. [Accessed 17 03 2016].

[20] W. I. o. T. W. I. P. M. Dónal McCarthy Telecommun. Software & Sytems Group, J.

Hange, K. Doyle, E. Robson, D. Conway, S. Ivanov, L. Radziwonowicz, R. Kleinfeld, T. Michalareas, T. Kastrinogiannis, N. Stasinos and F. Lampathaki, " Personal Cloudlets: Implementing a User-centric Datastore with Privacy Aware Access Control for Cloud-Based Data Platforms," *Technical and Legal aspects of data Privacy and Security, 2015 IEEE/ACM 1st International Workshop on,* pp. 38-43, 18 05 2015.

[21] P. M. J. H. K. D. E. R. D. C. S. I. Ł. R. R. K. T. M. T. K. N. S. F. L. Dónal McCarthy, "Privacy Aware Access Control for Cloud-Based Data Platforms," in *Cyber Security and Privacy*, vol. 530, Brussels, Springer International Publishing, 2015, pp. pp 26-37.

[22] G. M. P. Iosif Alvertis Decision Support Systems Laboratory National Technical University of Athens, F. Lampathaki, D. Askounis and T. Kastrinogiannis, "A community-based, Graph API framework to integrate and orchestrate cloud-based services," *IEEE/ACS 11th International Conference on Computer Systems and Applications (AICCSA),* pp. 485 - 492, 10-13 11 2014.

[23] S. O. (. M. P. Rodrigo Illera (LOG), "Security and Privacy Considerations for Cloud-based Services and Cloudlets," 31 01 2013. [Online]. Available: http://www.openi-ict.eu/wp-content/uploads/2013/11/OPENi_D2.3.pdf. [Accessed 21 04 2016].

[24] MyDex, "The Case for Personal Information Empowerment: The rise of the personal data store," [Online]. Available: https://mydex.org/wp-content/uploads/2010/09/The-Case-for-Personal-Information-Empowerment-The-rise-of-the-personal-data-store-A-Mydex-White-paper-September-2010-Final-web.pdf. [Accessed 03 04 2016].

[25] MyDex, "Mydex Developer Documentation," [Online]. Available: https://dev.mydex.org/fyi/open-source.html. [Accessed 12 04 2016].

[26] MyDex, "Members Account," [Online]. Available: https://pds.mydex.org/user/register. [Accessed 04 05 2016].

[27] C. Bizer, T. Heath and T. Berners-Lee, "Linked Data - The Story So Far," *International Journal on Semantic Web and Information Systems (IJSWIS).*

[28] W3C, "W3C Semantic Web Activity," [Online]. Available: https://www.w3.org/2001/sw/. [Accessed 04 05 2016].

[29] "Linked Data FAQ," [Online]. Available: http://structureddynamics.com/linked_data.html#question_14. [Accessed 04 05 2016].

[30] W3C, "Resource Description Framework (RDF)," [Online]. Available: https://www.w3.org/RDF/. [Accessed 04 05 2016].

[31] W. H. T. B.-L. N. Shadbolt, "The Semantic Web Revisited.," [Online]. Available: http://eprints.soton.ac.uk/262614/1/Semantic_Web_Revisted.pdf. [Accessed 04 05 2016].

[32] K. Nikolaos and S. Dimitrios-Emmanuel, Materializing the Web of Linked Data, Athens: Springer International Publishing, 2015.

[33] SIOC-Project, [Online]. Available: http://www.sioc-project.org/. [Accessed 20 04 2016].

[34] F. Sergio, G. Frédérick and I. Kingsley, "SIOC Ontology: Applications and Implementation Status," 15 05 2009. [Online]. Available: http://rdfs.org/sioc/applications/. [Accessed 18 05 2016].

[35] Openlink, "virtuoso-opensource," [Online]. Available: https://github.com/openlink/virtuoso-opensource. [Accessed 16 12 2015].

[36] Virtuoso Open Link, "Virtuoso Universal Server," [Online]. Available: http://virtuoso.openlinksw.com/. [Accessed 23 12 2015].

[37] AppScale Systems, "How AppScale implements the Google App Engine APIs,"

[Online]. Available: https://github.com/AppScale/appscale/wiki/How-AppScale-implements-the-Google-App-Engine-APIs. [Accessed 05 05 2016].

[38] C. B. S. P. C. K. N. M. S. S. R. W. Navraj Chohan, "AppScale: Scalable and Open AppEngine Application Development and Deployment".

[39] Internet Engineering Task Force (IETF), "The OAuth 2.0 Authorization Framework," [Online]. Available: https://tools.ietf.org/html/rfc6749. [Accessed 05 05 2016].

[40] TykTechnologies, [Online]. Available: https://tyk.io/. [Accessed 02 03 2016].

[41] TykTechnologies, "Tyk API Gateway," [Online]. Available: https://github.com/TykTechnologies/tyk. [Accessed 12 03 2016].

[42] A. Narayanan, S. Barocas, V. Toubiana, H. Nissenbaum and D. Boneh, "A Critical Look at Decentralized Personal Data Architectures," February 2012. [Online]. Available: http://randomwalker.info/publications/critical-look-at-decentralization-v1.pdf. [Accessed 18 04 2016].

[43] P. Ann Cavoukian, "Privacy by Design - The 7 Foundational Principles," [Online]. Available: https://www.ipc.on.ca/images/resources/7foundationalprinciples.pdf. [Accessed 27 04 2016].

[44] K. J. B. D. E. R. G. M. H. J. K. a. M. M. Shilton, "Designing the Personal Data Stream: Enabling Participatory Privacy in Mobile Personal Sensing.," 2009. [Online]. Available: http://ssrn.com/abstract=1999839. [Accessed 28 04 2016].

[45] Tyk Technologies, "Tyk API Gateway Benchmarks," [Online]. Available: https://tyk.io/tyk-api-gateway-benchmarks/. [Accessed 06 05 2016].

[46] K. M. D. Doyle, "OPENi White Paper: An End Users Perspective: Digital Identity Putting the Genie Back in the Bottle," 09 2014. [Online]. Available: http://www.openi-ict.eu/wp-content/uploads/2014/07/openi_whitepaper.pdf. [Accessed 28 04 2016].

[47] E. R. (. G. M. (. D. C. (. J. H. (. Dónal McCarthy (WIT), "OPENi Cloudlet Framework Design Document," 23 09 2014. [Online]. Available: http://www.openi-ict.eu/wp-content/uploads/2014/10/OPENi_D3.5.pdf. [Accessed 21 04 2016].

[48] C. Krintz, "The AppScale Cloud Platform Enabling Portable, Scalable Web Application Deployment," 03 05 2013.

Appendices

I. API Documentation

The documentation was prepared using Apiary and can be found here http://docs.iupds.apiary.io/#. The file can also be found in the attached archive file.

II. License

Non-exclusive license to reproduce thesis

I, Sunday Ayanbode Ayandokun

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to:

1.1. reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until the expiry of the term of validity of the copyright, and

1.2. make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until the expiry of the term of validity of the copyright,

Application-agnostic Personal Storage for Linked Data,
supervised by Peep Küngas, PhD,

2. I am aware of the fact that the author retains these rights.

3. I certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu, **19.05.2016.**