Design Document

Archival Service

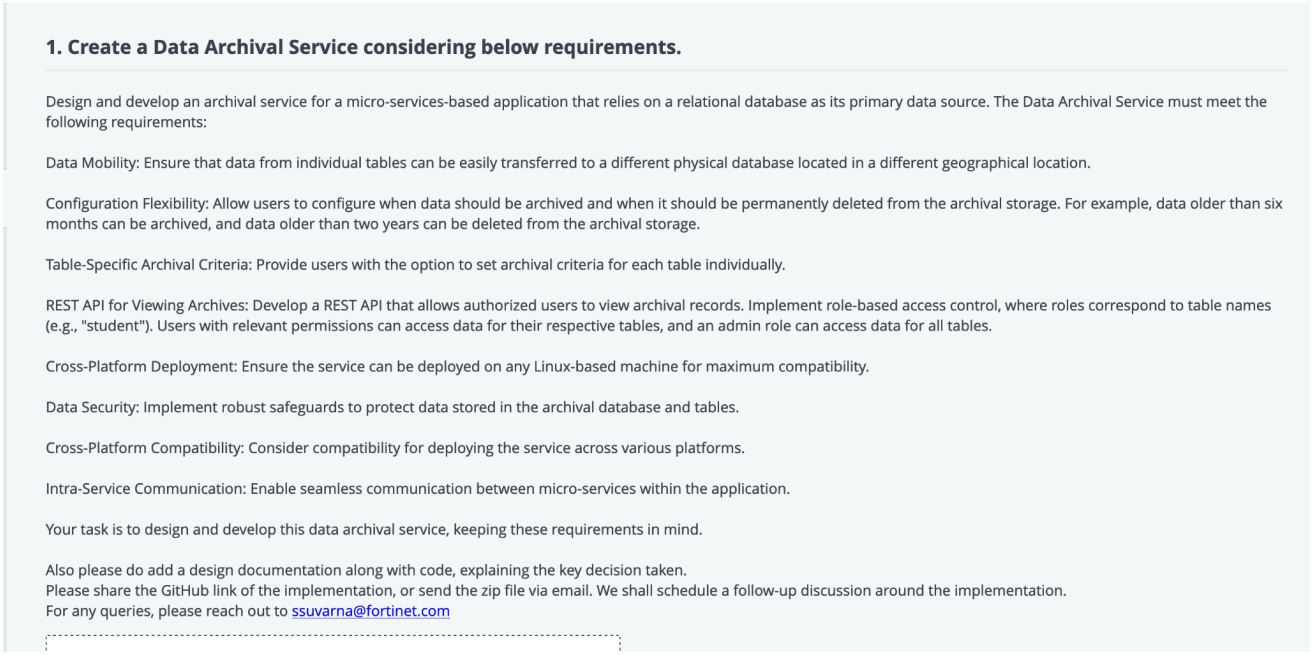
Written By	Rahul Prajapati
Reviewed By	
Version	1.0.0

Table of Contents

Introduction

This project is about relational database table archives to local or global geographical locations and deletion on archive from archive database .

Requirements



1. Create a Data Archival Service considering below requirements.

Design and develop an archival service for a micro-services-based application that relies on a relational database as its primary data source. The Data Archival Service must meet the following requirements:

Data Mobility: Ensure that data from individual tables can be easily transferred to a different physical database located in a different geographical location.

Configuration Flexibility: Allow users to configure when data should be archived and when it should be permanently deleted from the archival storage. For example, data older than six months can be archived, and data older than two years can be deleted from the archival storage.

Table-Specific Archival Criteria: Provide users with the option to set archival criteria for each table individually.

REST API for Viewing Archives: Develop a REST API that allows authorized users to view archival records. Implement role-based access control, where roles correspond to table names (e.g., "student"). Users with relevant permissions can access data for their respective tables, and an admin role can access data for all tables.

Cross-Platform Deployment: Ensure the service can be deployed on any Linux-based machine for maximum compatibility.

Data Security: Implement robust safeguards to protect data stored in the archival database and tables.

Cross-Platform Compatibility: Consider compatibility for deploying the service across various platforms.

Intra-Service Communication: Enable seamless communication between micro-services within the application.

Your task is to design and develop this data archival service, keeping these requirements in mind.

Also please do add a design documentation along with code, explaining the key decision taken.
Please share the GitHub link of the implementation, or send the zip file via email. We shall schedule a follow-up discussion around the implementation.
For any queries, please reach out to ssuvarna@fortinet.com

Assumptions

Archive service (web based restful service) is a centralized application for microservices based architecture relational database table archival.
Communication between source database server and archival database(destination database server) is enabled. (SSH and HTTP).
Frontend part is not covered in the system. User can interact with Restful services.

Solution overview

We are going to use HTTP and SSH protocol to execute the whole execution smoothly.
To manage archive restful service and for archive table and load to destination db will configure agents.

We have to archive the relation db from various microservices db to some different geographical location.
Application provide 5 different modules to interact with it.
User module
Service catalog
Source agent
Destination agent
Execution
User module will have admin role and normal user role. Admin can add user and assign role to user (role is table name as per given req).

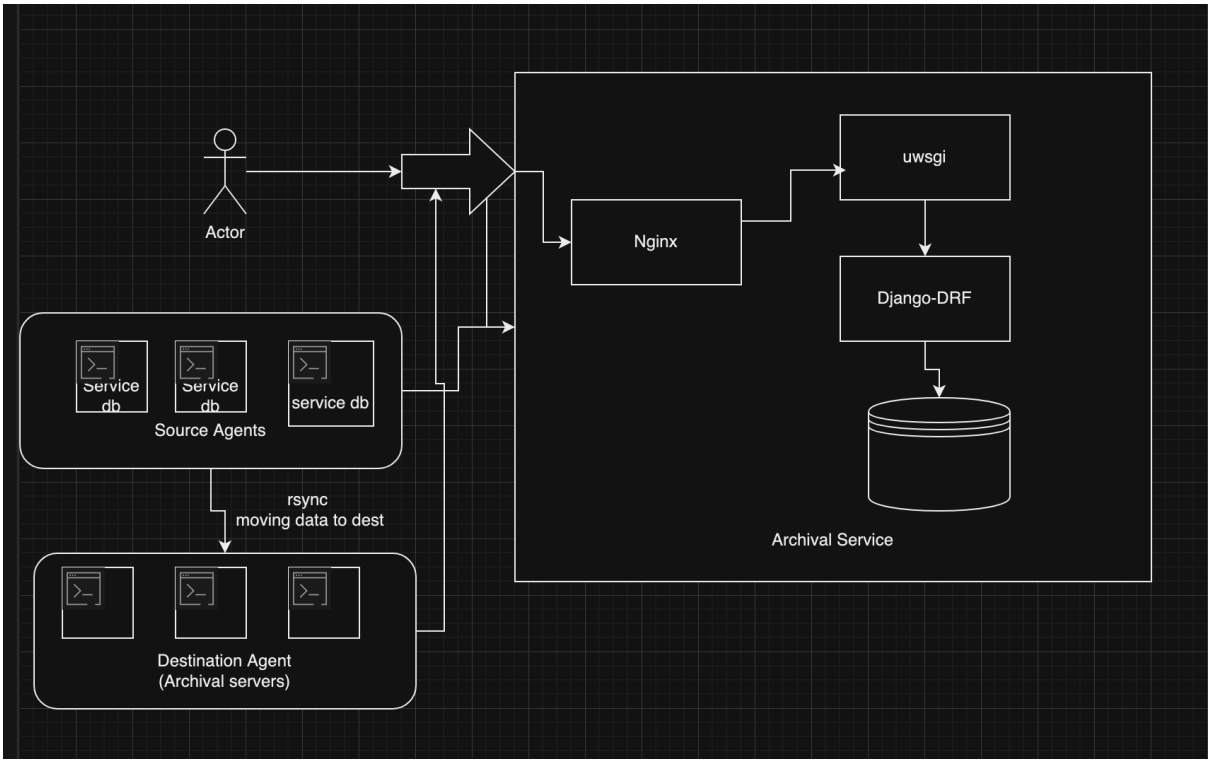
Admin or Agents will configure the services in system with the help of Service catalog which include registration of new service and only admin can add new service.Registration process include Microservice detail, services server, db schema detail -(table/partition), destination server details, archive period, and purging period, schedule etc.
Assigned role user so user also can configure/update the archival policies for authorised table.
Agent- We introduce two agent for archive one is source agent and other one is destination agent. These are crons which will execute on source servers and destination servers.
Source agent will fetch data from archive api in order to archive the table based on the configuration.
After getting all archive details, it will generate records in metadb Execution table with unique id per table archive. Then it will dump each table with that unique id, copy data to destination server's temp location, delete the archive data from all microservices and update the status.
Destination agent will look in temp location, if any archive is present then with the help of unique id, it will fetch detail for database/schema/table from API. Load the data to archive database and update the status of archived in metadb execution table.
Destination agent will have once more task for deleting the data. It will fetch details from metadb and all table it will read data, if deletion occur then again create a task in deletion execution table and update the status.

Tech stack

Python’s django rest framework: in order to perform different operation and interacting with system. The reason to choose django is it is faster, good support, scalable and good inbuilt support different level of auth and user level permissions etc
Uwsgi - To run django application will use uwsgi as application server. It has a large request handling capacity and easy to scale.
Nginx: As web server in order to serve any static content, rate limiting and further if we want to have multiple microservices then can be used for api gateway and load balancing.
rsync: To copy archival to destination. It fast compare to scp.
Postgresql: To store our meta data for services and users management. Easy to scale and we can configure cluster based nodes.
Docker: To deploy the service anywhere across different system.
Git: Code management system.

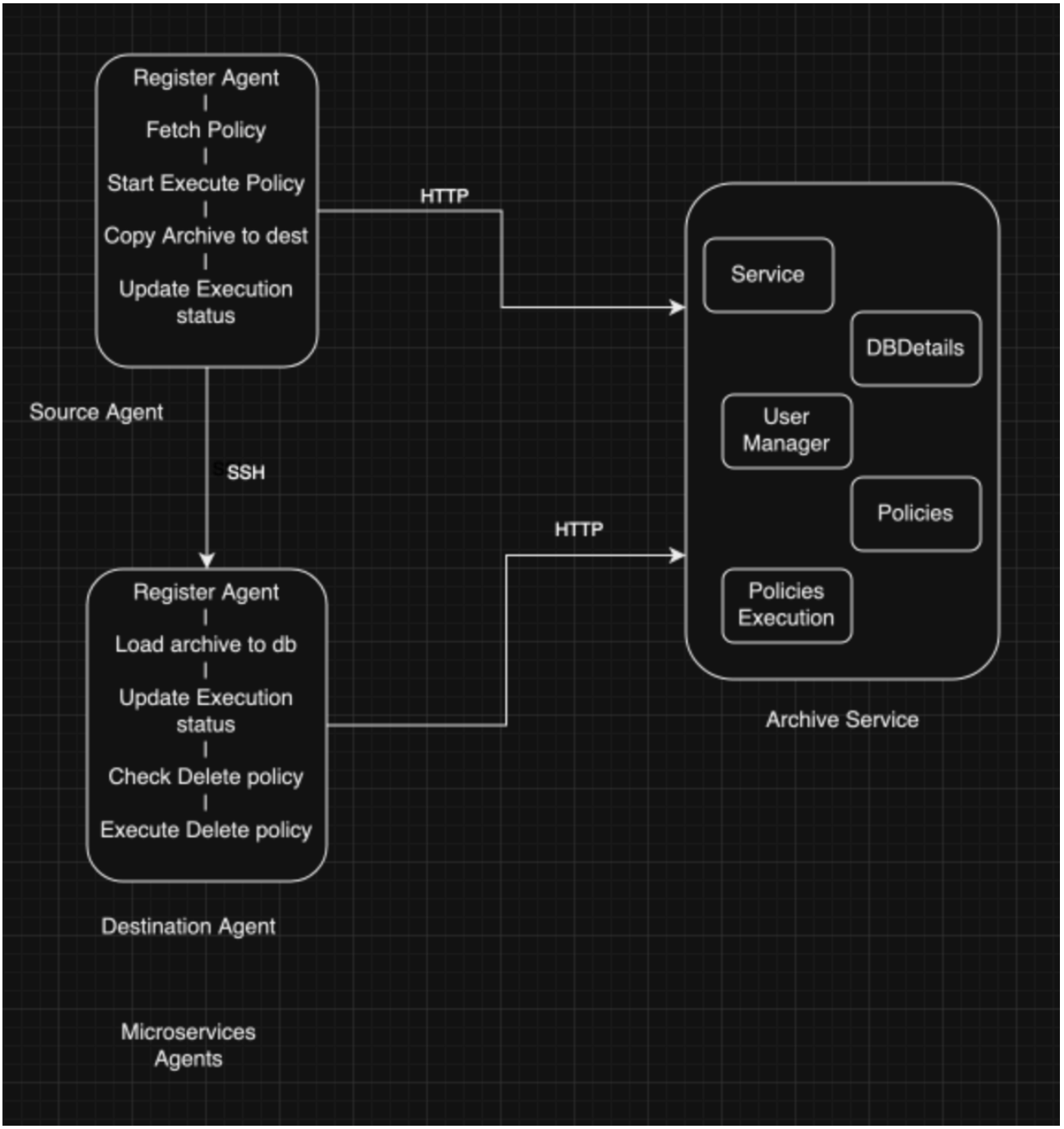
High level architecture

Agent will communication with archive service and in between communication to copy archive.



Agent service communications

Agent is actually performing archival and deletion and Archive Service is managing agent via Restful services agent.



Agent workflow

Source Agent:

Source agents responsible for archival of table.
Cron will be scheduled on source db server based on policy configured on archival service.

Startup Flow:
1) Login to source db and get all schema details: tables/ partitions/ users etc
2) Call Archive service catalog api's to register agents details i.e
api/v1/service/add, api/v1/dbserver/add, api/v1/table/add, api/v1/schema/add etc

Execution Flow:
1) Retrieve policies information, and get all list of policy to be executed for archive.
2) If policies is ready to execute then it will create record in the execution table to track the execution of archive
and get the unique id for all policies to be executed.
3) Once policy is retrieved, then it start archival of tables and rename the dump with unique id.
To archive fetch the detail how old data should be archived. Archieve option 1) Select query
and it will dump the data to some temp location or we can create partition for data and copy the data files of
table and rename data file to unique id.
4) Once archive is ready then it will copy all files to destination server(destination details we can get from archive policy)
It will copy data to temp location on destination server.
5) Update the status of archive in execution table as source_success/source_failed
6) Authorised user can check the status of policy execution and configure policy.

Destination Agent:

Destination agents responsible for process archival to destination database table, and deletion of tables configured in policy.
Cron will be scheduled on destination db server based on policy configured on archival service.

Execution Flow:

Archive Process:

1) Check the temp folder where all the archive is placed and get list of archive and call api /execution/<uid>

get the policy info from it.

2) Get table and process the archive.

3) Update the status of archive in execution table as failed/completed.

Deletion

1) Retrieve policies information, and get all list of policy to be executed for deletion.

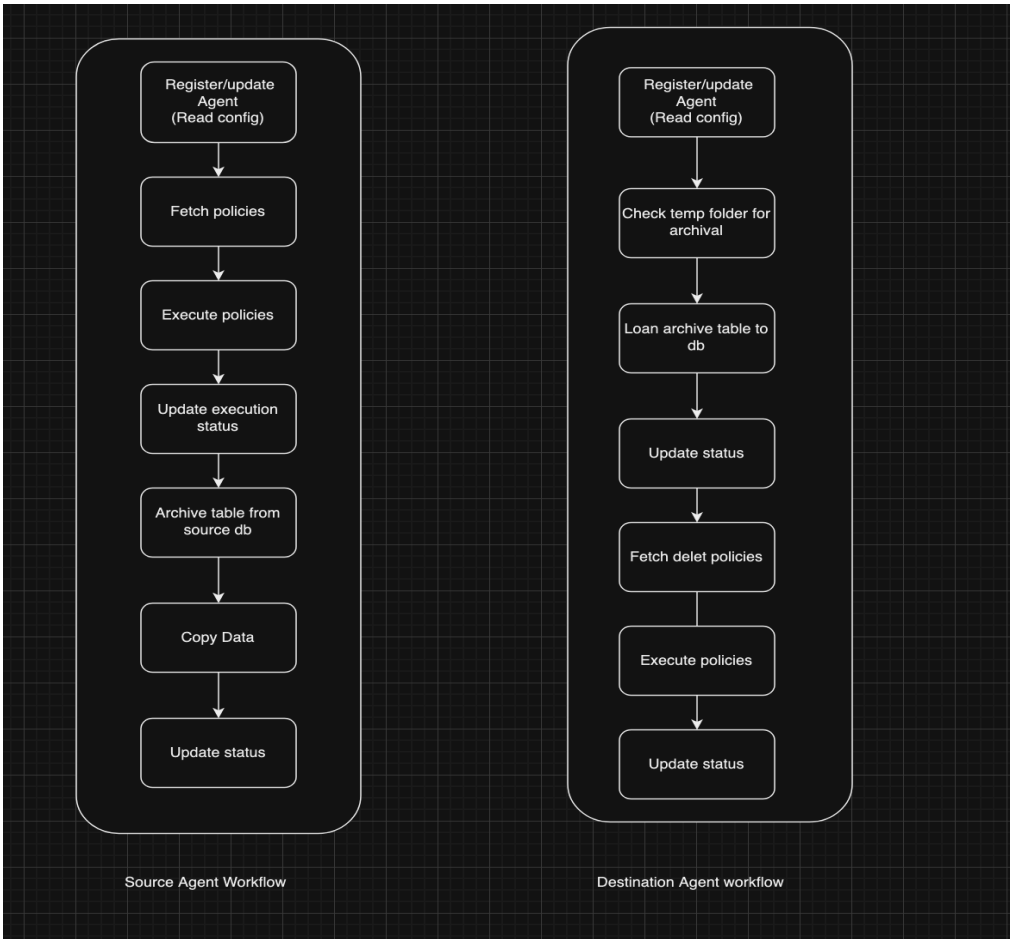
2) If policies is ready to execute then it will create record in the execution table to track the execution of archive

and get the unique id for all policies to be executed.

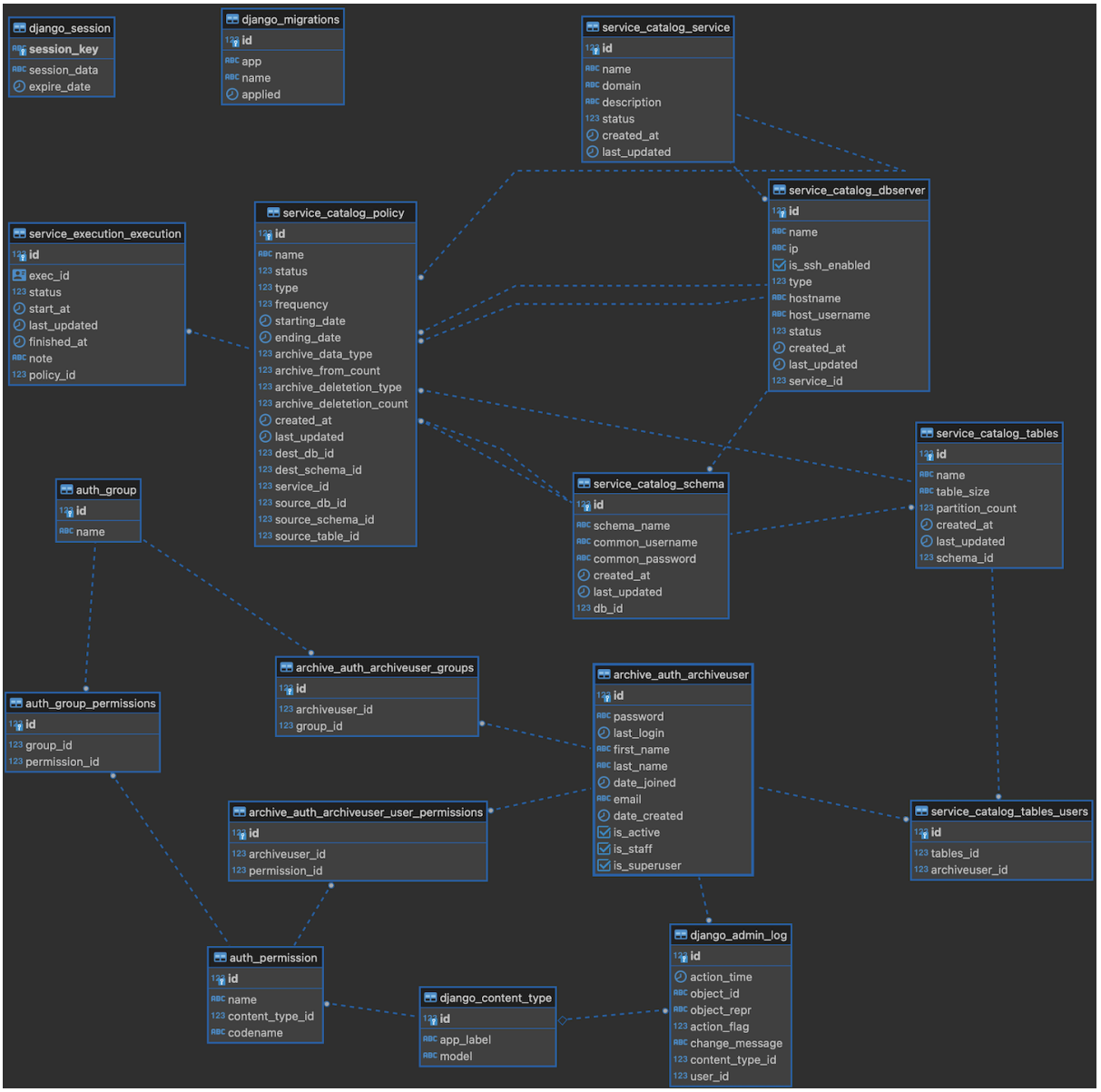
3) Once policy is retrieved, then it start deletion of tables and entry all deletion table name in execution archive service.

5) Update the status of archive in execution table.

6) Authorised user can check the status of policy execution and configure policy.



Schema design



Archive Service Userflow

Admin User

1) Manage Other user (JWT auth implemented)
2) Manage Service
3) Manage DB
4) Register Tables and Schema
5) Manage Policies
6) Manage Policy Execution

Normal User

1) Created and Manage policies for authorised tables
2) Check execution for policies (Archival and Deletion)

API - Restful services

service
api/v1/service/add/
api/v1/service/list/
servicapi/v1/e/update/<int:pk>/
api/v1/service/patch/<int:pk>/
api/v1/service/delete/<int:pk>/

dbserver
api/v1/dbserver/add/
api/v1/dbserver/list/
api/v1/dbserver/update/<int:pk>/
api/v1/dbserver/patch/<int:pk>/
api/v1/dbserver/delete/<int:pk>/

schema
api/v1/schema/add/
api/v1/schema/list/
api/v1/schema/update/<int:pk>/
api/v1/schema/patch/<int:pk>/
api/v1/schema/delete/<int:pk>/

table
api/v1/table/add/
api/v1/table/list/
api/v1/table/update/<int:pk>/
able/patch/<int:pk>/

api/v1/table/delete/<int:pk>/

policy
api/v1/policy/add/
api/v1/policy/list/
api/v1/policy/update/<int:pk>/
api/v1/policy/patch/<int:pk>/
api/v1/policy/delete/<int:pk>/

execution
api/v1/execution/add/
api/v1/execution/list/
api/v1/execution/update/<int:pk>/
api/v1/execution/patch/<int:pk>/
api/v1/execution/delete/<int:pk>/

User managment
api/v1/auth/register1/
api/v1/auth/register/
api/v1/auth/register/<int:pk>/
api/v1/auth/login/
api/v1/auth/token/refresh/
api/v1/auth/token/verify/