






OVERVIEW \

intro to service-engine

Generalized service engine that auto provisions REST, GraphQL & gRPC services that support CRUD operations with full validation to tables, views and materialized views of several popular databases

Key Features

- Auto provision of server resources to support CRUD in REST, GraphQL & gRPC with minimal configuration
- Validation at the source
- Support DB Schema migration
- Auto Generate API Documentation (openapi3 docs, GraphQL Playground, .proto file)
- Provide a hook for intercepting the query before execution in order to append extra data (specifically to support partition keys)



General Service Engine
gRPC & GraphQL & REST

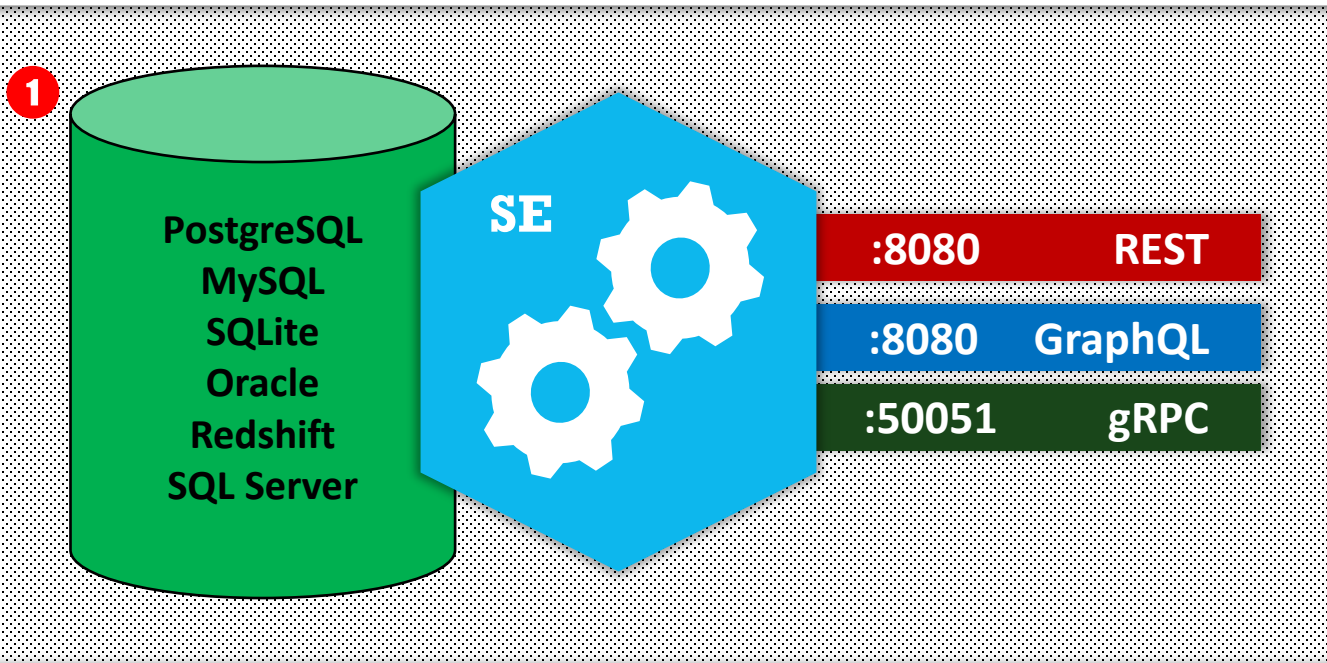
Multiple Implementations:

- 1 [service-engine](#)
 - SRC for [NPM package](#)
- 2 [service-engine-docker](#)
 - Implements [NPM package](#)
 - SRC for [Generic Docker Image](#)
- 3 [service-engine-template](#)
 - Implements [Generic Docker Image](#)
 - Ideal for bootstrapping new projects

HOW IT WORKS \

startup sequence

1. Run DB Migrations
2. Autodetects DB resources via DB Survey
3. Builds JOI validators for all DB resources
4. Publishes REST, GraphQL & gRPC services



```
2
```

```
gear .env
1 DB_CLIENT=pg
2 DB_HOST=data.some_domain.com
3 DB_PORT=5432
4 DB_USER=some_user
5 DB_PASSWORD=secret
6 DB_DATABASE=some_database
7 PAGINATION_LIMIT=100
```

```
3
```

```
docker run --rm -d \
  --env-file ../.env \
  -p 8080:8080 \
  -p 50051:50051 \
  sudowing/service-engine
```

- 4 ## Key REST Endpoints
- Health Check Route
<http://localhost:8080/healthz>
 - OpenAPI3 Definitions
<http://localhost:8080/openapi>
 - .proto
<http://localhost:8080/proto>
 - GraphQL Playground
<http://localhost:8080/some-app-service/graphql/>

KEY CONCEPT \

service call → structured query language

1

```
http://localhost:8080/sample-app-name/service/${schema}_${table}/
? occupation = engineer
& state.in = NJ|PA
& handle.like = sudo%
& page = 5
& limit = 30
& orderBy = handle,name_last:desc
& fields = id,handle,email,name_first
& separator = |
```

2

```
select
  id
  , handle
  , email
  , name_first
from
  public.some_table -- REST call made to /public_some_table
where
  occupation = 'engineer'
and
  state in ('NJ', 'PA')
and
  handle like 'sudo%'
order by
  handle
  , name_last desc
limit 30
offset 120
```

3

REST API Endpoint Patterns

action	method	endpoint pattern	note
CREATE	POST	/service/:resource	1 or many
READ	GET	/service/:resource/record?field_a=alpha&field_b=bravo	if resource is keyed
UPDATE	PUT	/service/:resource/record?field_a=alpha&field_b=bravo	if resource is keyed
DELETE	DELETE	/service/:resource/record?field_a=alpha&field_b=bravo	if resource is keyed
READ	GET	/service/:resource	many supported operators

4

REST API Headers

req header key	req header value	resp header key	resp header value description
		x-request-id	UUID assigned to request for injection into log
x-get-sql	truthy	x-sql	SQL built by service
x-get-count	truthy	x-count	unpaginated count for submitted query

5

SQL Operators

field. operator	sql operator	multiple seperated args	# of args
field	= (default)	false	
field. equal	=	false	
field. gt	>	false	
field. gte	>=	false	
field. lt	<	false	
field. lte	<=	false	
field. not	<>	false	
field. like	like	false	
field. null	is null	false	
field. not_null	is not null	false	
field. in	in (...values)	true	
field. not_in	not in (...values)	true	
field. range	between x and y	true	2
field. not_range	not between x and y	true	2
field. geo_bbox	geo_bbox	true	4
field. geo_radius	geo_radius	true	3
field. geo_polygon	geo_polygon	false	

6

Additional Query Context

key	description
page	Pagination Page
limit	Pagination Limit
fields	Fields to return from the SQL query
orderBy	Fields to order results by. (field_a:desc,field_b,field_c:desc)
separator	Separator used to separate values submitted in request

SYSTEM DESIGN \

normalize → validate → build SQL

REST
view

GraphQL
resolver

gRPC
method

2
service call normalizer

4
query validation

5
query builder

7
database

1

```
http://localhost:8080/sample-app-name/service/${schema}_${table}/  
? occupation = engineer  
& state.in = NJ|PA  
& handle.like = sudo%  
& page = 5  
& limit = 30  
& orderBy = handle,name_last:desc  
& fields = id,handle,email,name_first  
& separator = |
```

3

```
{  
  "payload": {  
    "occupation": "engineer",  
    "state.in": "NJ|PA",  
    "handle.like": "sudo%",  
  },  
  "context": {  
    "page": 5,  
    "limit": 30,  
    "orderBy": "handle,name_last:desc",  
    "fields": "id,handle,email,name_first",  
    "separator": "|"  
  },  
}
```

6

```
select  
  id  
  , handle  
  , email  
  , name_first  
from  
  public.some_table -- REST call made to /public_some_table  
where  
  occupation = 'engineer'  
  and  
  state in ('NJ', 'PA')  
  and  
  handle like 'sudo%'  
order by  
  handle  
  , name_last desc  
limit 30  
offset 120
```

DOCUMENTATION \

video feature overviews & requirements

Application Considerations

Unsupported Characters in GraphQL

All DB schema names, resource names and field names must adhere to GraphQL Schema Definition Language (SDL)

DB Permissions

Migration support is optional -- however if you want to use it you'll need to ensure the service account being used by the app has appropriate permissions to create objects and write records.

Returning Fields on CREATE & UPDATE

Not all DBs support returning fields on INSERT & UPDATE statements.

Postgres does and it's the recommended engine for new projects implemented this library.

For example, MySQL & Sqlite3 return 201s with no-body in REST and other payloads in GraphQL & gRPC.

1 ## Feature Overview Videos

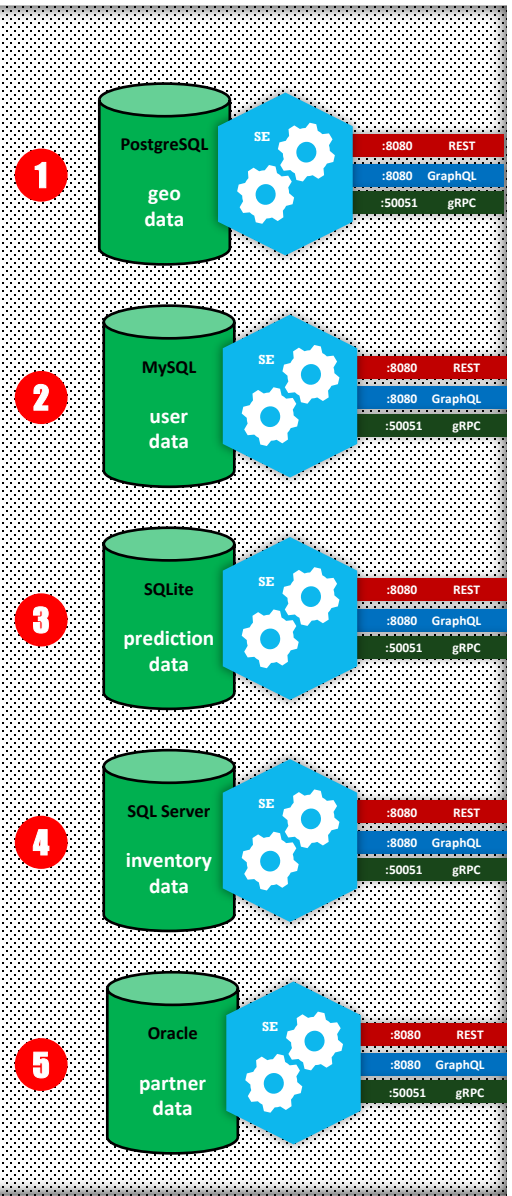
- [Quick Start](#)
- [Key REST Endpoints](#)
- [Insomnia Import](#)
- [CRUD Operations](#)
- [SQL Operators](#)
- [Query Context](#)
- [API Response Metadata](#)
- [Debug Mode](#)
- [Permissions](#)
- [DB Schema Migrations](#)
- [GraphQL Playground and Geoqueries](#)
- [gRPC Service \(CRUD & Geoqueries\)](#)
- [Complex Resources \(subqueries & aggregate queries\)](#)
- [Middleware & Redactions](#)

CLOSING NOTES \

feedback, closing thought & contact info



Joe Wingard
github [@sudowing](#)
keybase [@sudowing](#)
linkedIn [@joewingard](#)




Project Feedback

Feedback and Recommendations are best received as [Pull-Requests](#) & [GitHub Issues](#).

Closing Thought

I hope this project is useful to you and your team. If you find it valuable, please send me a note!



api.domain.com

[/v1/geography](#)
[/v1/user](#)
[/v2/prediction](#)
[/v1/inventory](#)
[/v1/partner](#)

6