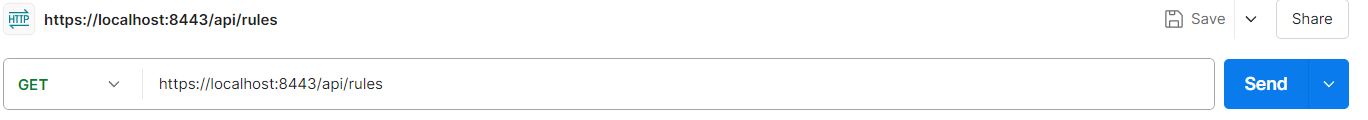
Explaining the application:  
The application for Zetta Online task consists of several API endpoints.

GET /api/rules  
returns one of the subclasses of AbstractRule.



Returns:  
  
{

"success": **true**,

"message": "Request Succesful",

"data": [

{

"id": 16,

"setName": "set1",

"rules": [

{

"type": "simple",

"id": 6,

"name": "r1",

"field": "age"

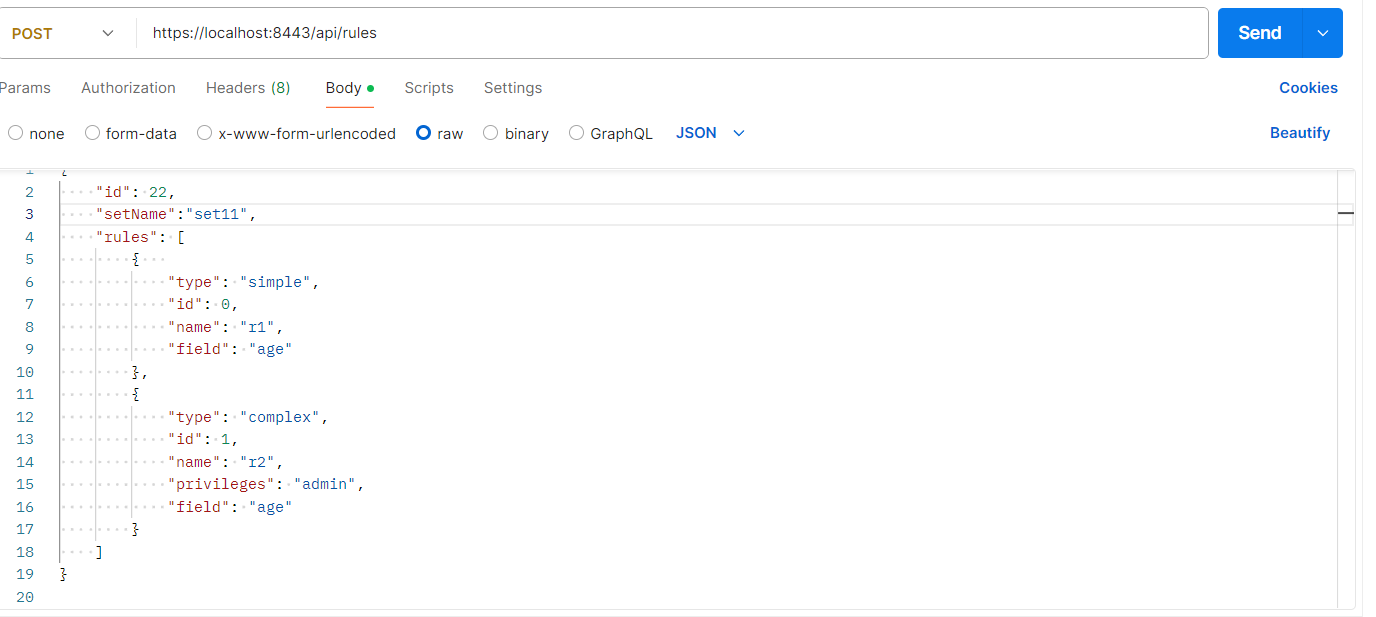
}

]

}

]

}

Post /api/rules:  
Adds a new rule:  
  
Takes such a request:

{

"id": 0,

"setName":"set1",

"rules": [

{

"type": "simple",

"id": 0,

"name": "r1",

"field": "age"

},

{

"type": "complex",

"id": 1,

"name": "r2",

"privileges": "admin",

"field": "age"

}

]

}

Returns such a request if ruleset exists or has duplicate rules:  
{

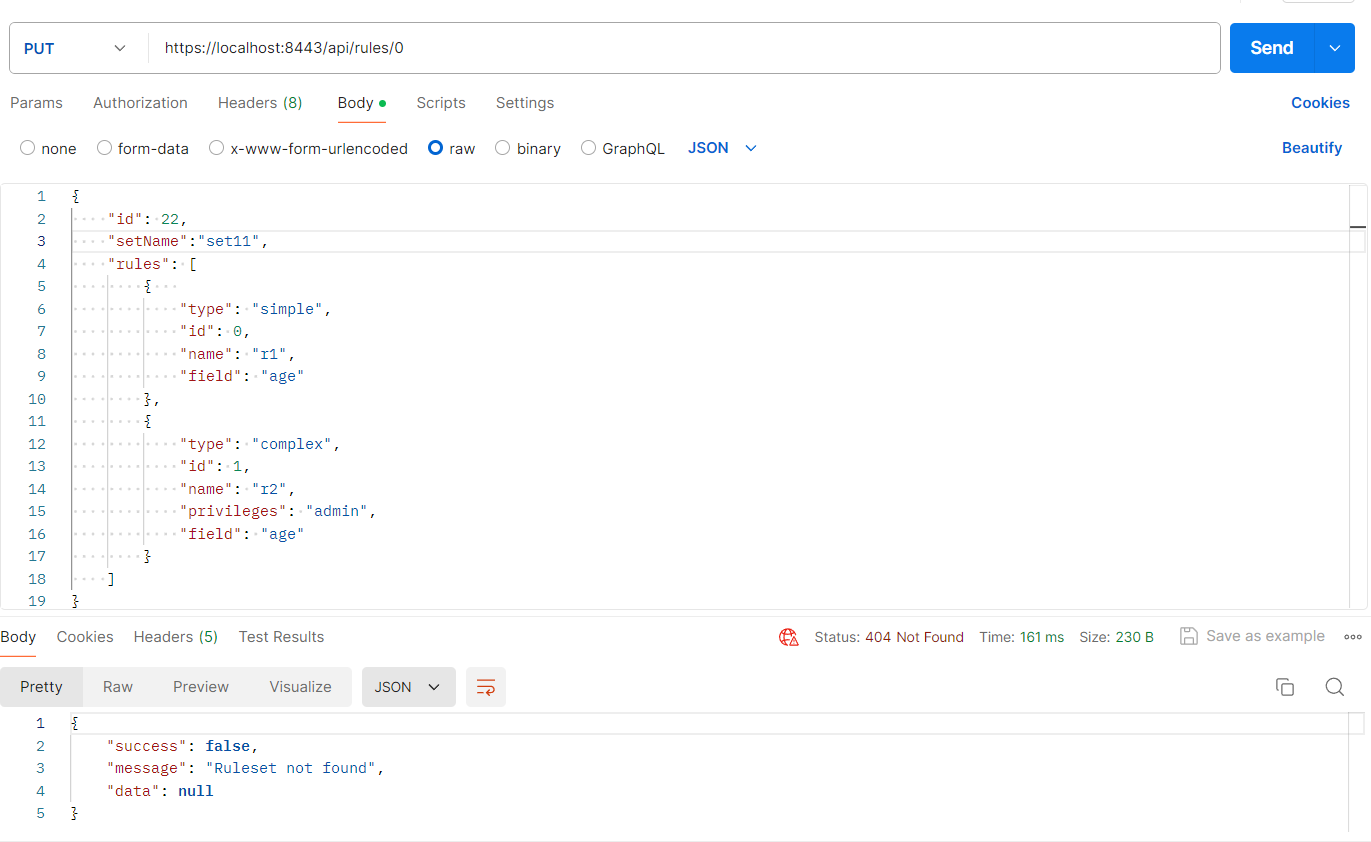
"success": **false**,

"message": "Ruleset Already Exists",

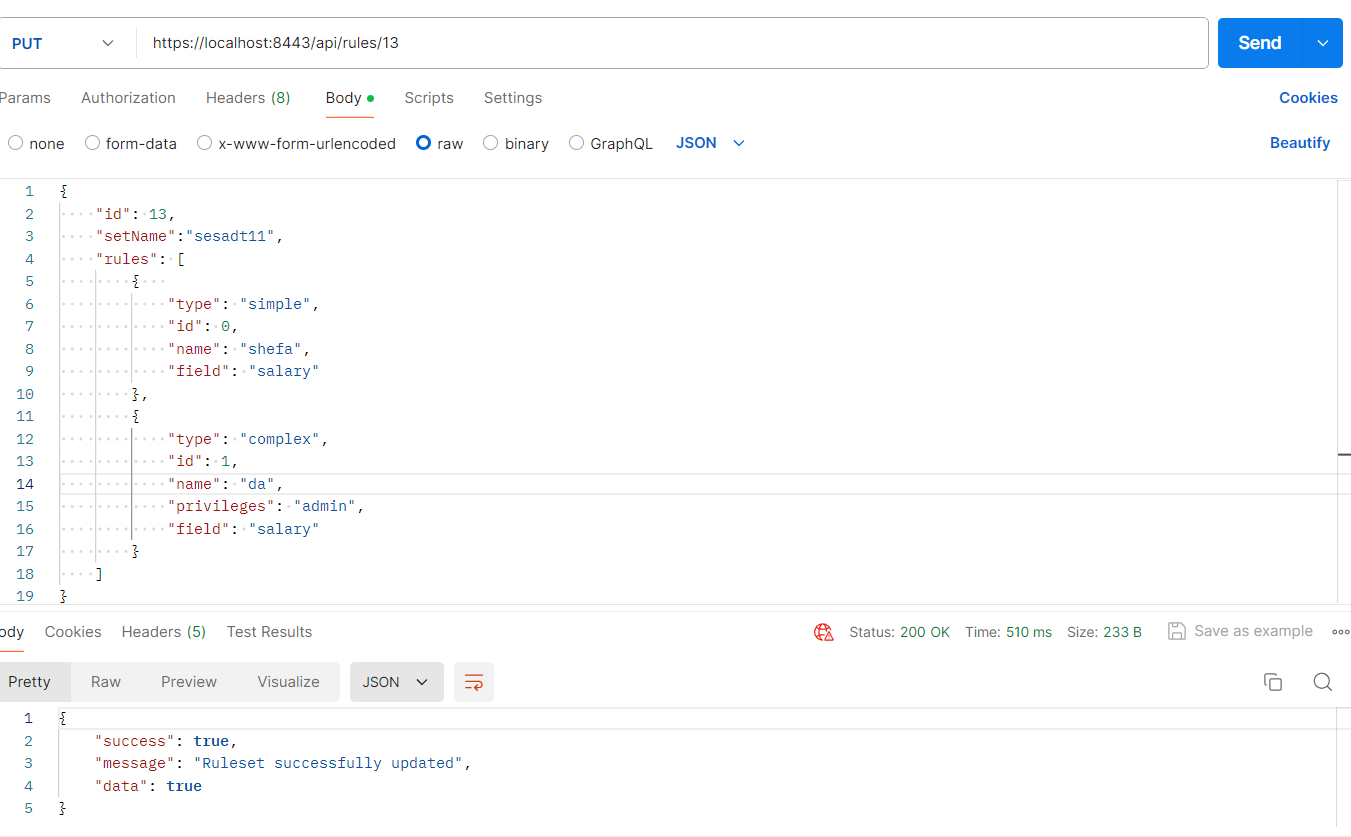
"data": **null**

}

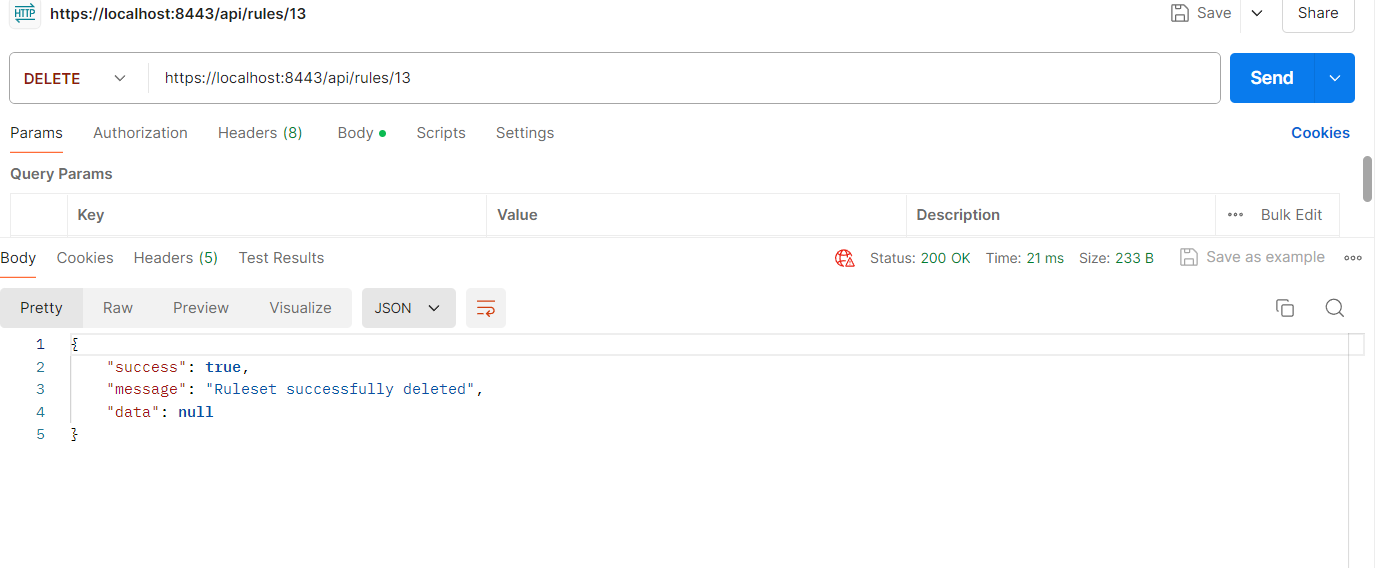
Put /api/rules/0

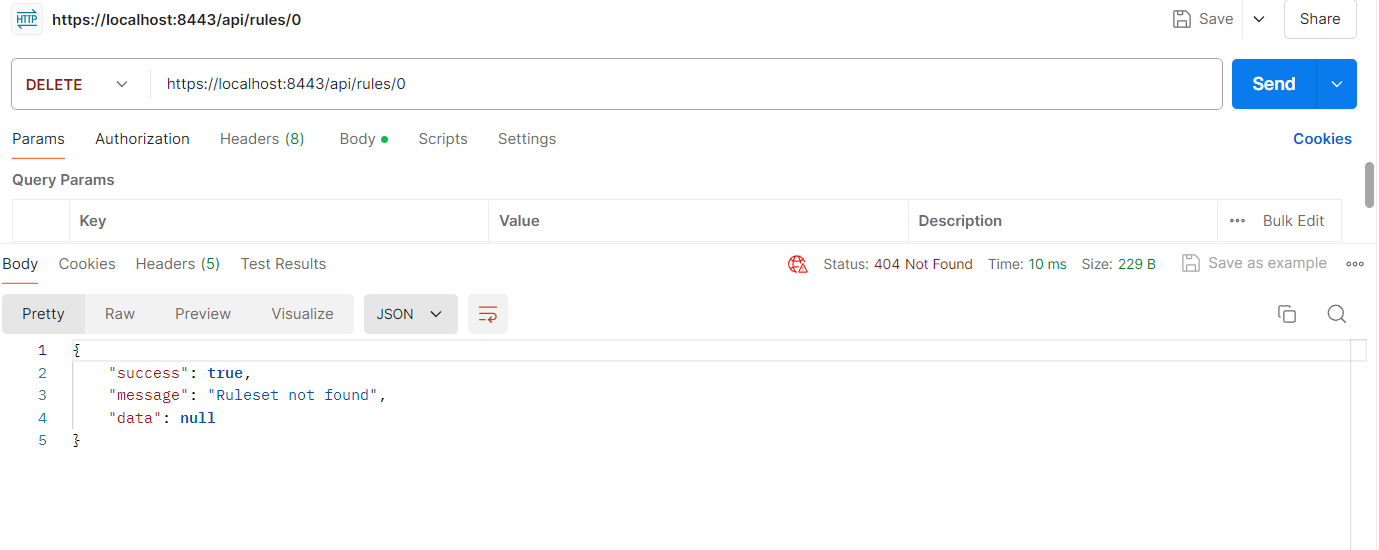


gives such a response if it is not found.

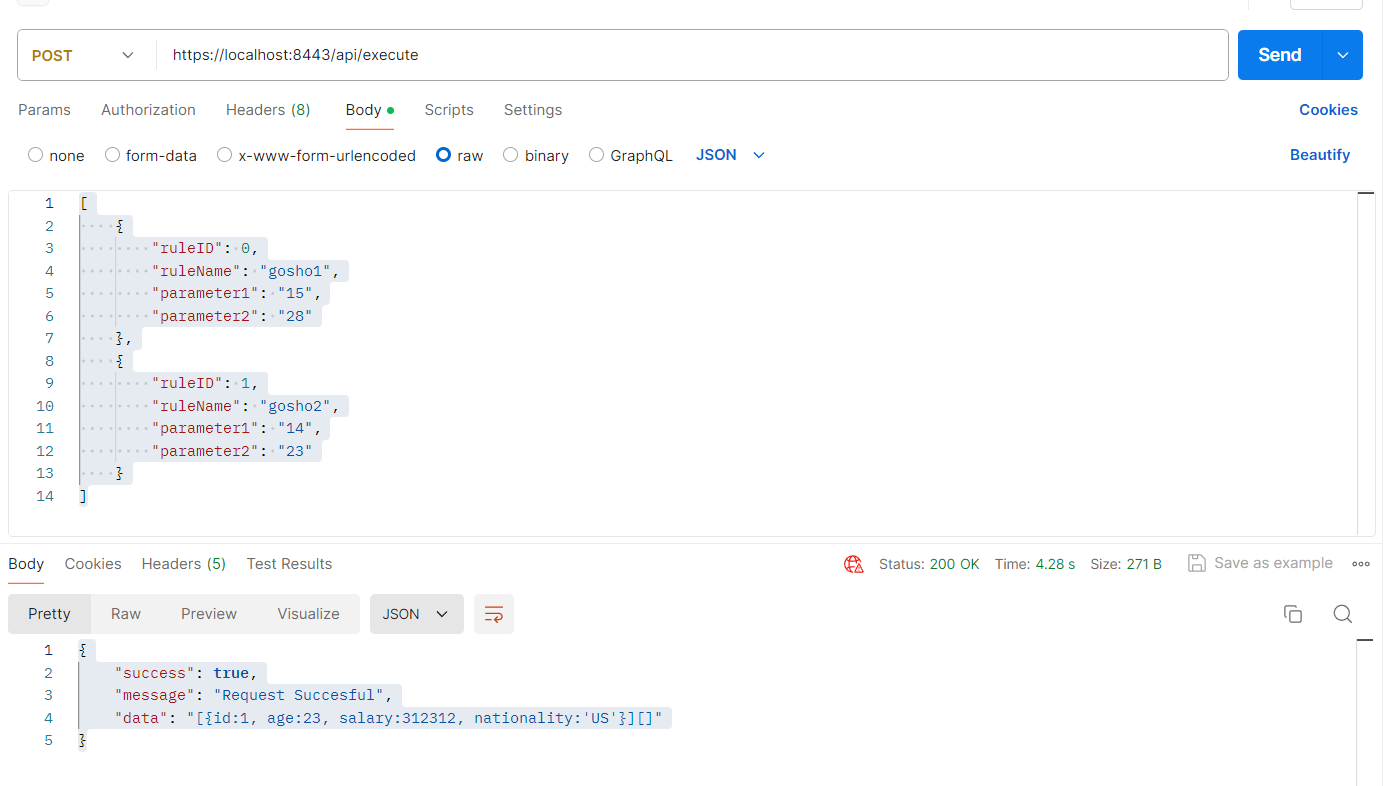


Gives such a response if ruleset is updated.

Delete:  
  
If succesfull

  
If not found.

Post /api/execute:  
Executes rules related to SQL database. Returns UserEntity class instance.



Takes such a request:

[

{

"ruleID": 0,

"ruleName": "gosho1",

"parameter1": "15",

"parameter2": "28"

},

{

"ruleID": 1,

"ruleName": "gosho2",

"parameter1": "14",

"parameter2": "23"

}

]

Returns such response:  
{

"success": **true**,

"message": "Request Succesful",

"data": "[{id:1, age:23, salary:312312, nationality:'US'}][]"

}

To run the project:  
Create postgreSQL database, i could conteinerize and send it but I don’t have docker installed.   
Run the following scripts:  
  
-- Database: userbase

-- DROP DATABASE IF EXISTS userbase;

CREATE DATABASE userbase

WITH

OWNER = postgres

ENCODING = 'UTF8'

LC\_COLLATE = 'English\_United Kingdom.1252'

LC\_CTYPE = 'English\_United Kingdom.1252'

LOCALE\_PROVIDER = 'libc'

TABLESPACE = pg\_default

CONNECTION LIMIT = -1

IS\_TEMPLATE = False;

Create table:

-- Table: public.users

-- DROP TABLE IF EXISTS public.users;

CREATE TABLE IF NOT EXISTS public.users

(

id bigint NOT NULL DEFAULT nextval('users\_id\_seq'::regclass),

age integer NOT NULL DEFAULT 0,

nationality character varying(2) COLLATE pg\_catalog."default",

salary bigint NOT NULL DEFAULT 1000,

CONSTRAINT users\_pkey PRIMARY KEY (id)

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public.users

OWNER to postgres;

-- Table: public.rules

-- DROP TABLE IF EXISTS public.rules;

CREATE TABLE IF NOT EXISTS public.rules ( id integer NOT NULL GENERATED ALWAYS AS IDENTITY ( INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ), rule\_type character varying(31) COLLATE pg\_catalog."default" NOT NULL, name character varying(255) COLLATE pg\_catalog."default" NOT NULL, privileges character varying(255) COLLATE pg\_catalog."default", nationality character varying(255) COLLATE pg\_catalog."default", field character varying(255) COLLATE pg\_catalog."default" NOT NULL, foreignkey integer, CONSTRAINT rules\_pkey PRIMARY KEY (id), CONSTRAINT rules\_name\_key UNIQUE (name), CONSTRAINT ukasd57j4i2nr2043y2jvq4m1fh UNIQUE (name), CONSTRAINT rules\_foreignkey\_fkey FOREIGN KEY (foreignkey) REFERENCES public.rulesetmodel (id) MATCH SIMPLE ON UPDATE NO ACTION ON DELETE NO ACTION )

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public.rules OWNER to postgres;

---------- ANOTHER TABLE ---------------

-- Table: public.rulesetmodel

-- DROP TABLE IF EXISTS public.rulesetmodel;

CREATE TABLE IF NOT EXISTS public.rulesetmodel ( id integer NOT NULL GENERATED ALWAYS AS IDENTITY ( INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ), set\_name character varying(255) COLLATE pg\_catalog."default", CONSTRAINT rulesetmodel\_pkey PRIMARY KEY (id), CONSTRAINT uk22bfgxu1xnapf1yv71395n4hk UNIQUE (set\_name) )

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public.rulesetmodel OWNER to postgres;

run the application jar in the “out” folder or run it from intellij.