PostGresConf : Dev Track
South Africa
8 Oct 2019

Karel van der Walt

ACPAS

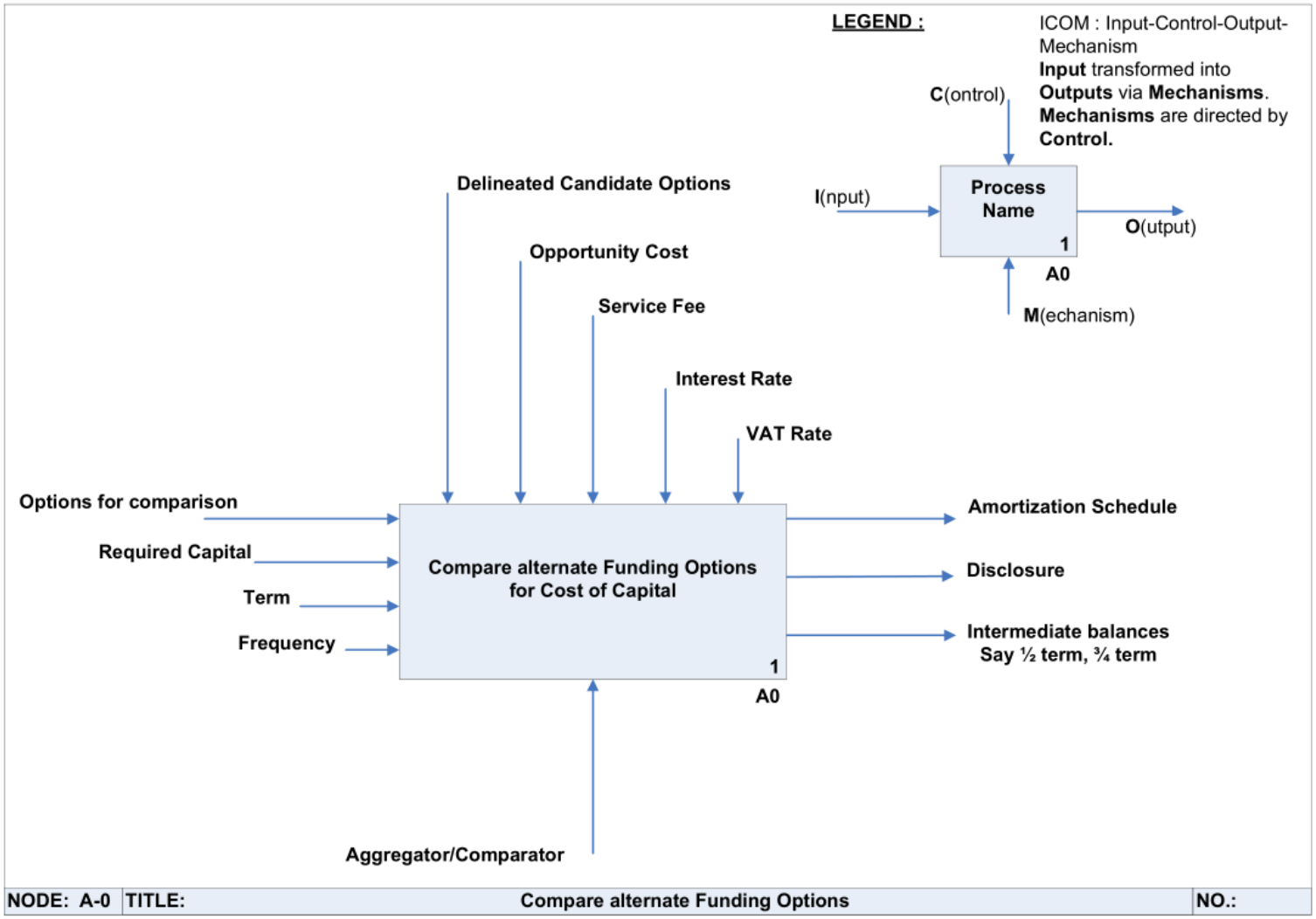
From models to hosted OpenAPI Specification (OAS)

Rapid OpenAPI with PostgreSQL, JSON, PostgREST, Swagger UI & NGINX

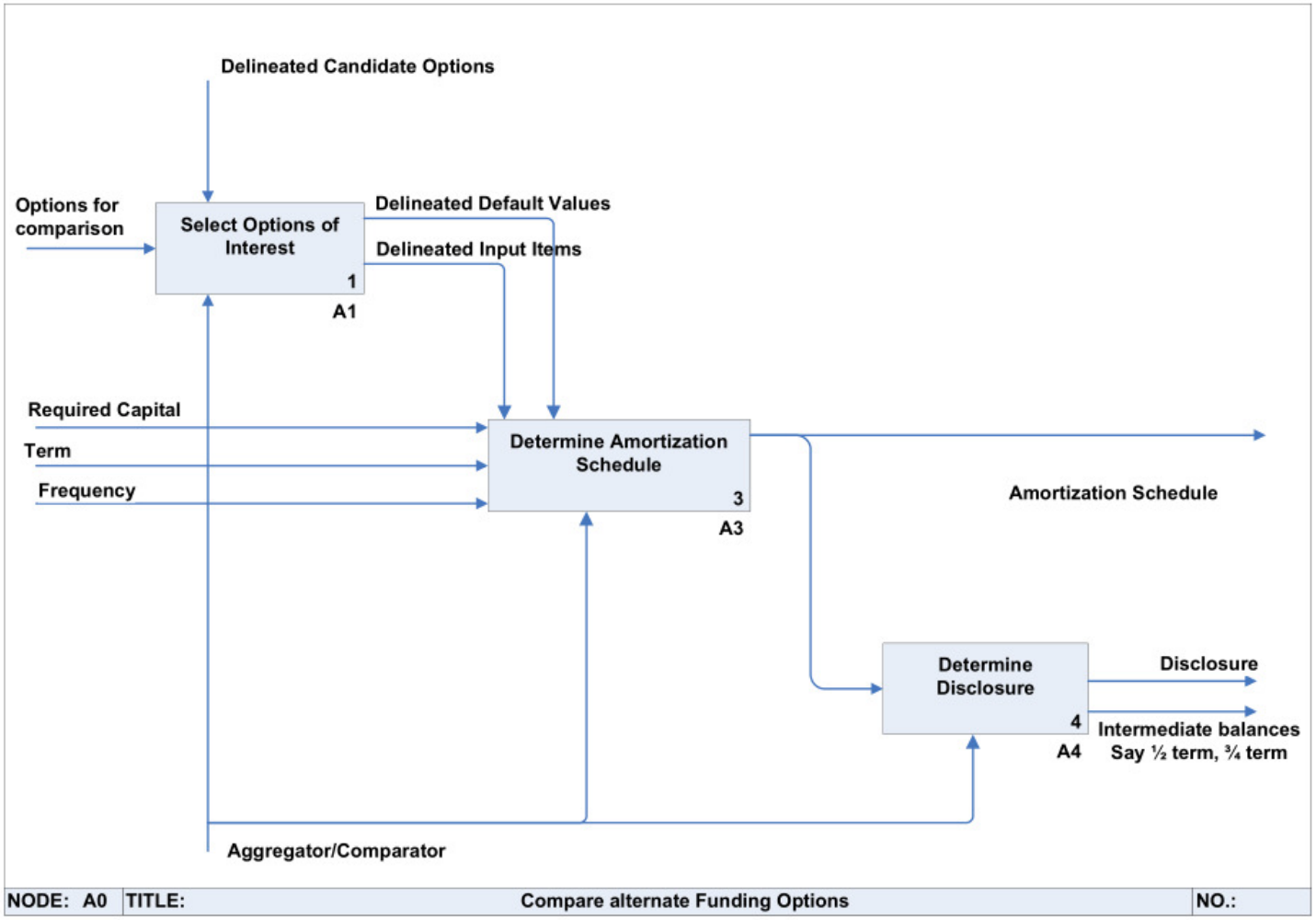
Agenda

- Realistic (‘non-trivial’) problem
- The data model (‘opinionated’)
- The process model (‘opinionated’)
- The REST API with minimal fuss
- Q&A (5 min)

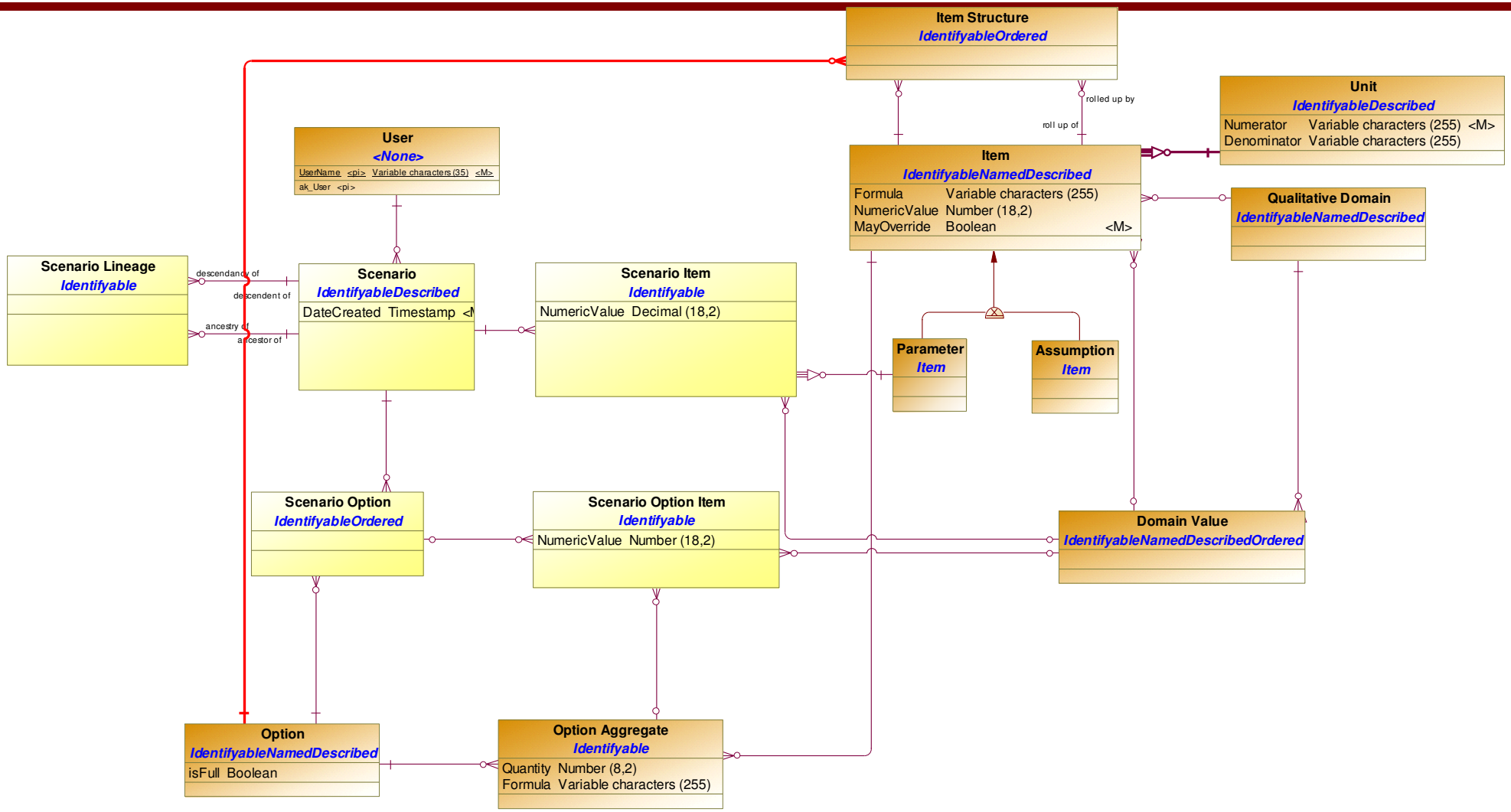
Context Diagram – Cost of Capital



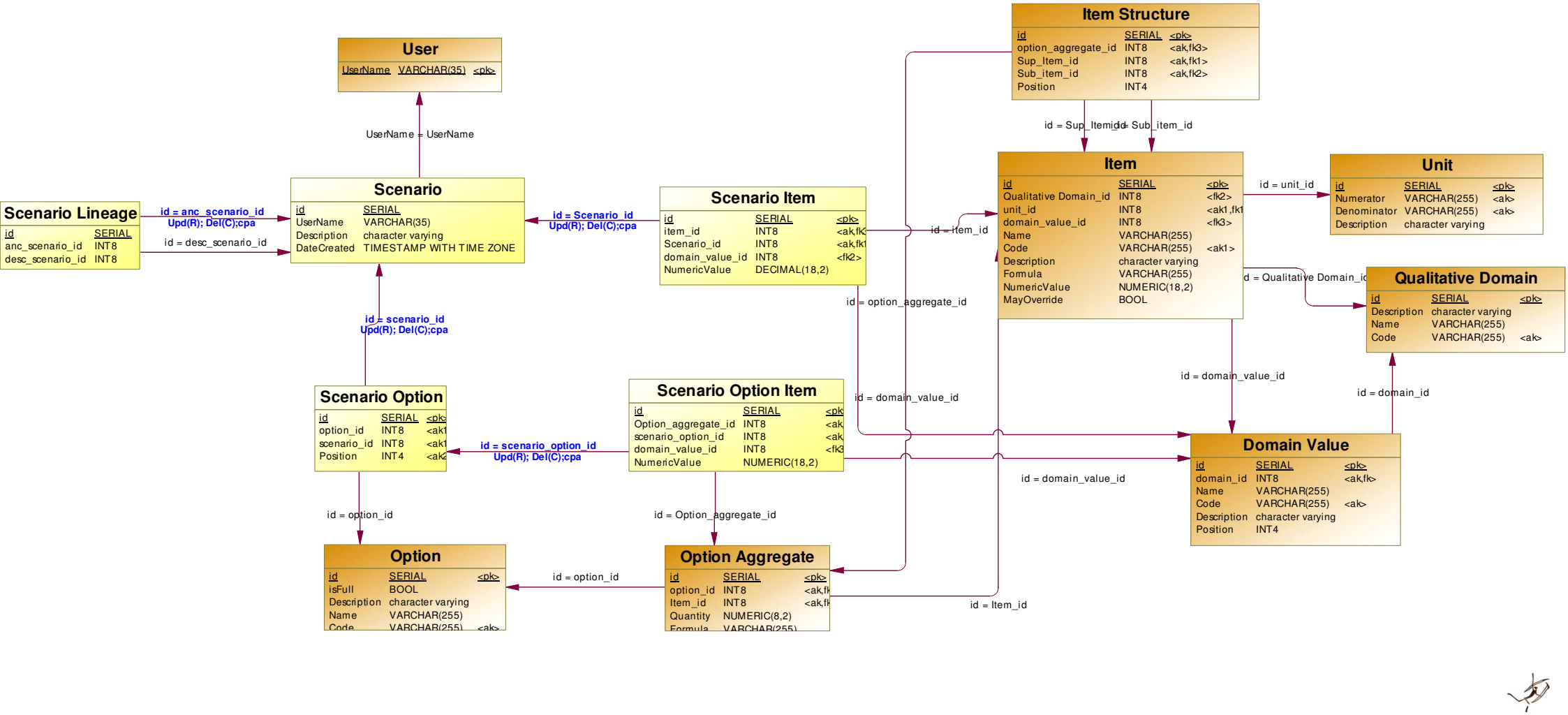
Context Diagram – Cost of Capital



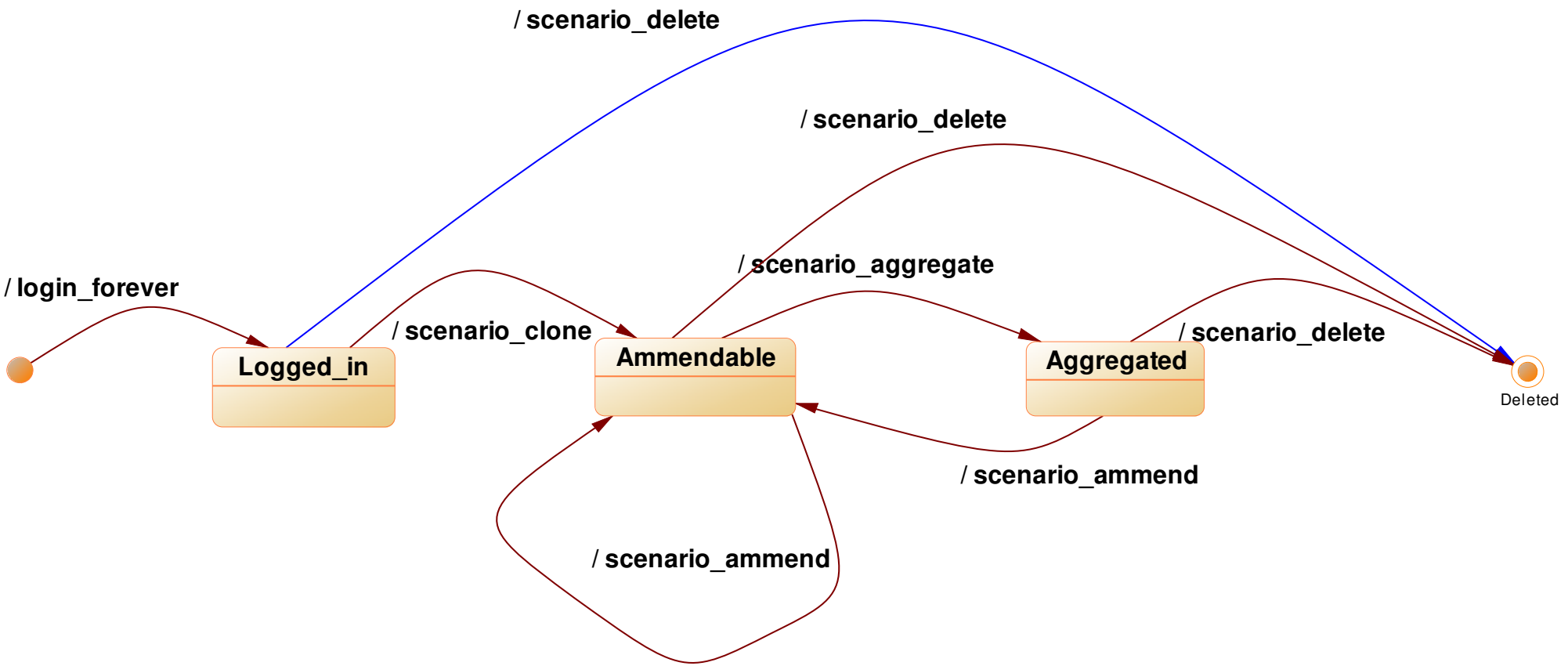
Conceptual Data Model



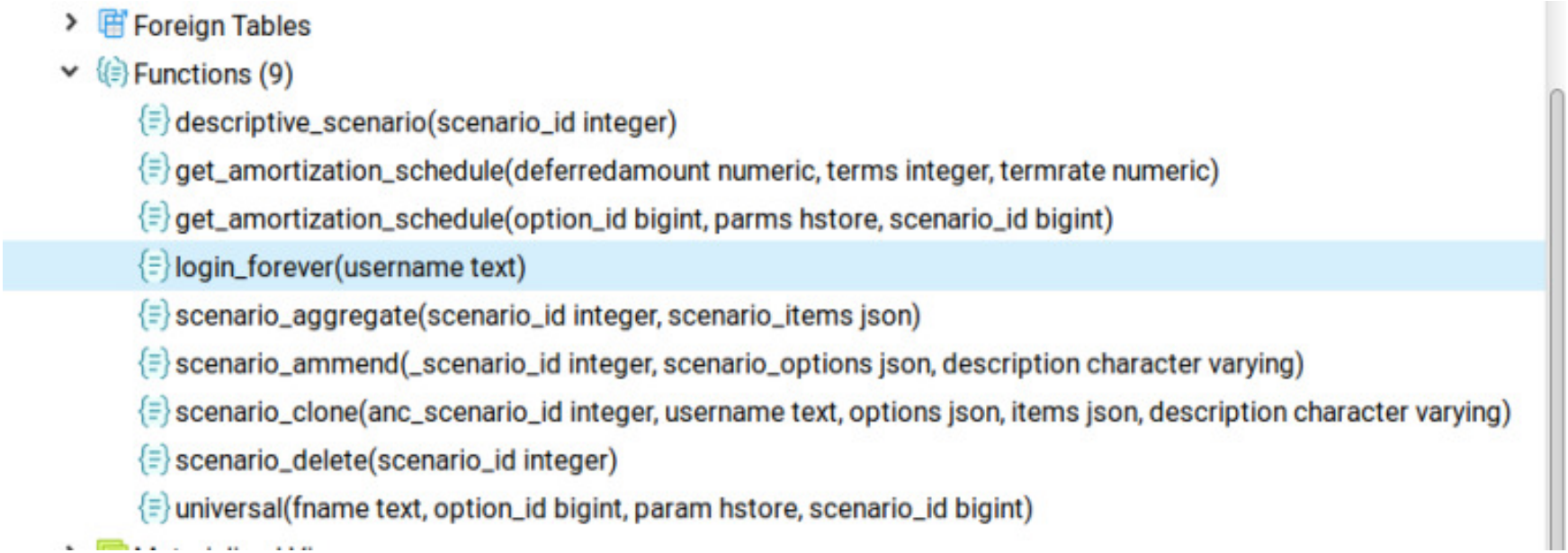
Physical Data Model



LifeCycle - Scenario

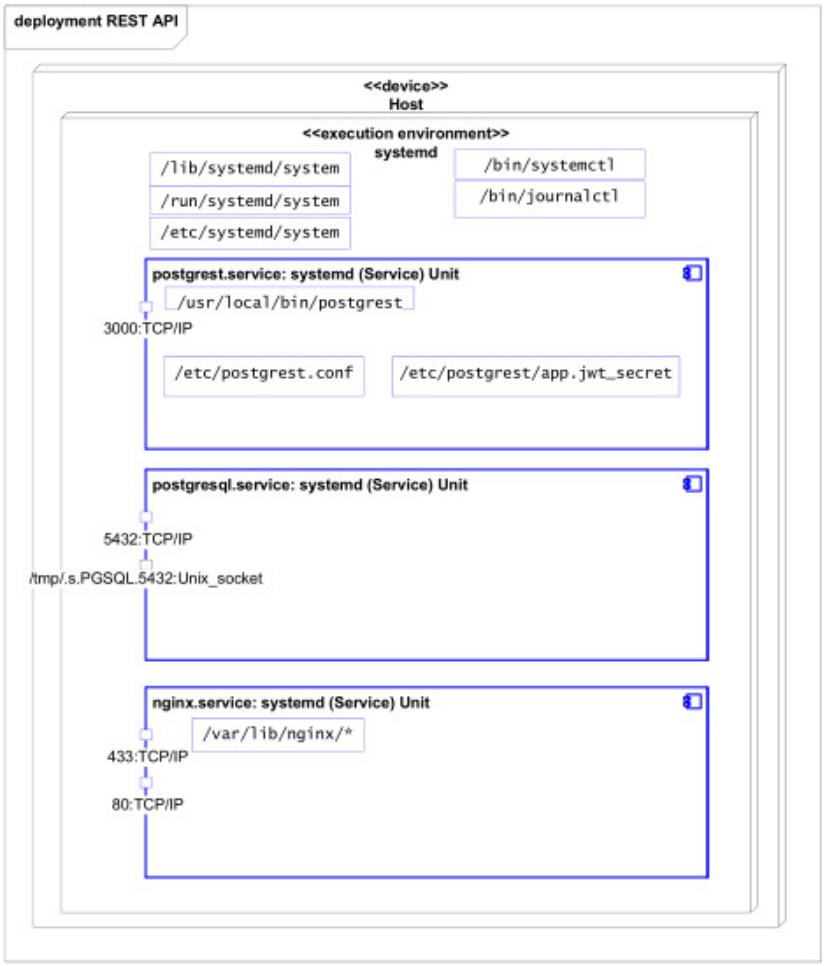


Enacting the Lifecycle – PL/pgSql

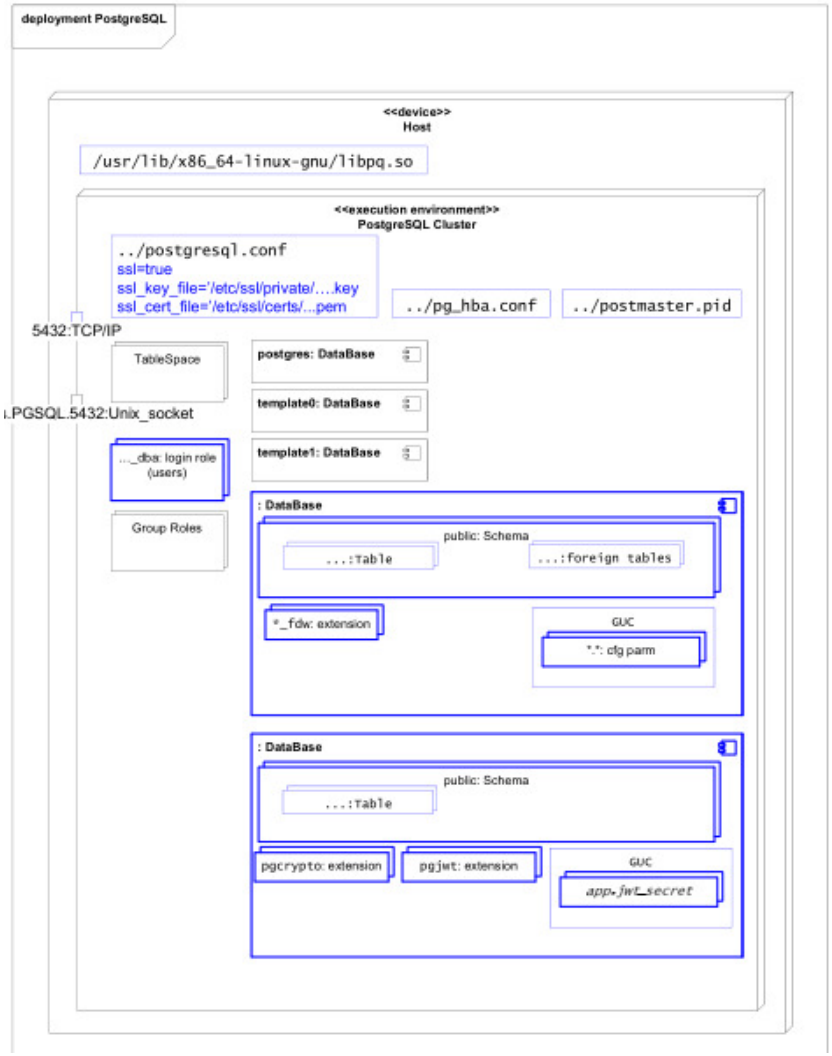


REST API with minimal fuss

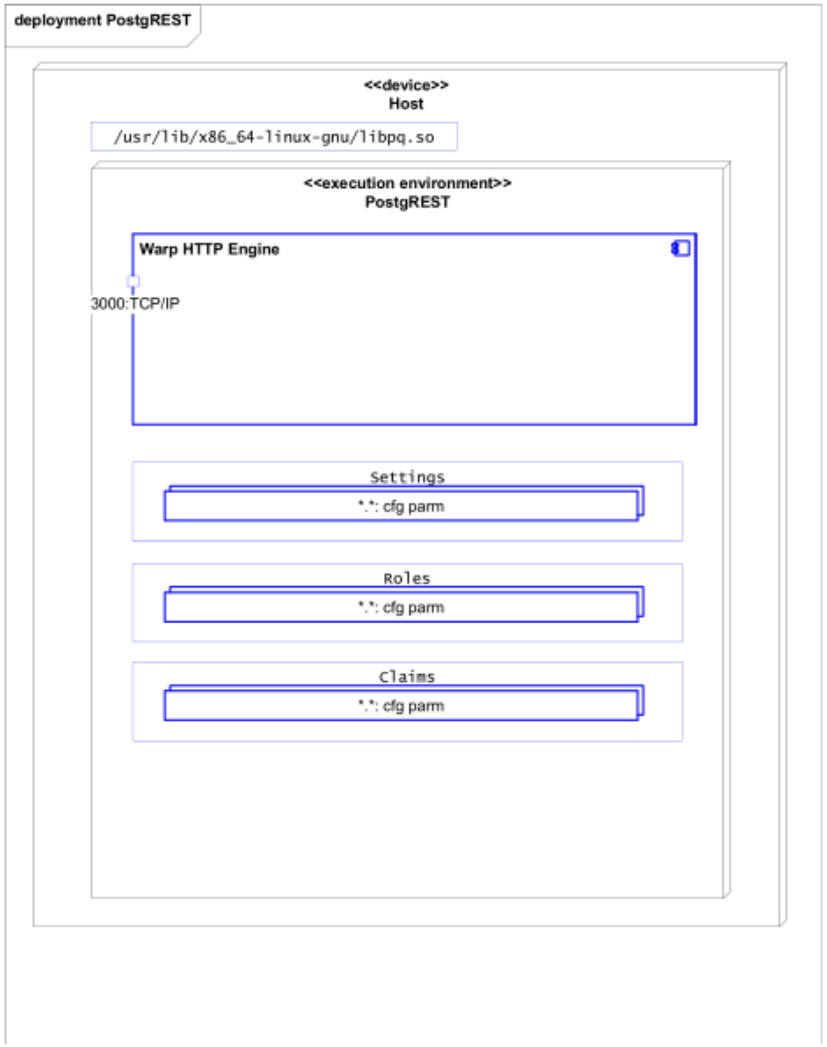
Intermezzo – Technical Architecture



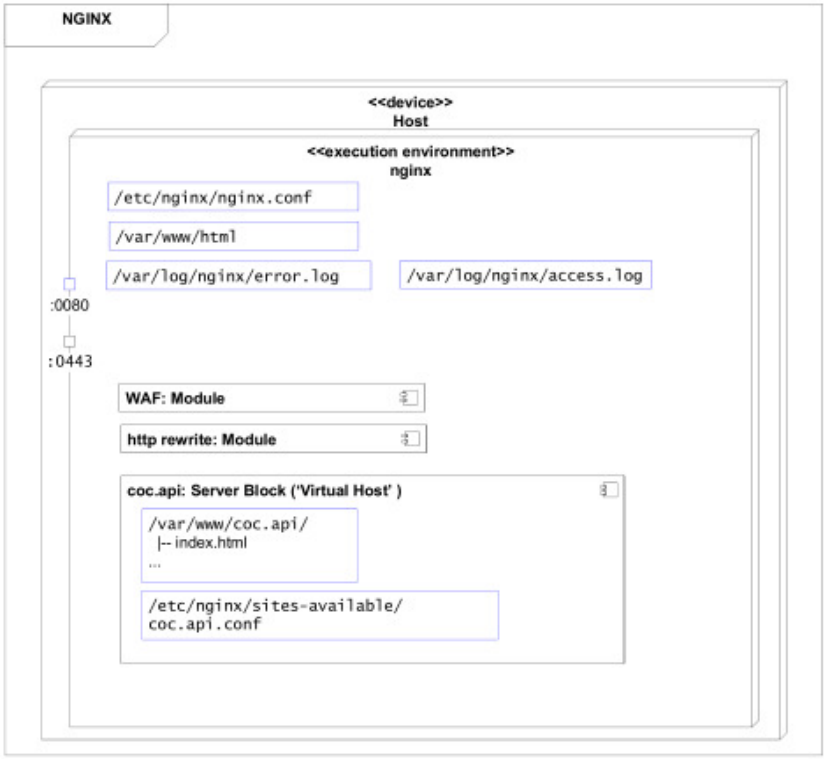
Intermezzo – Technical Architecture



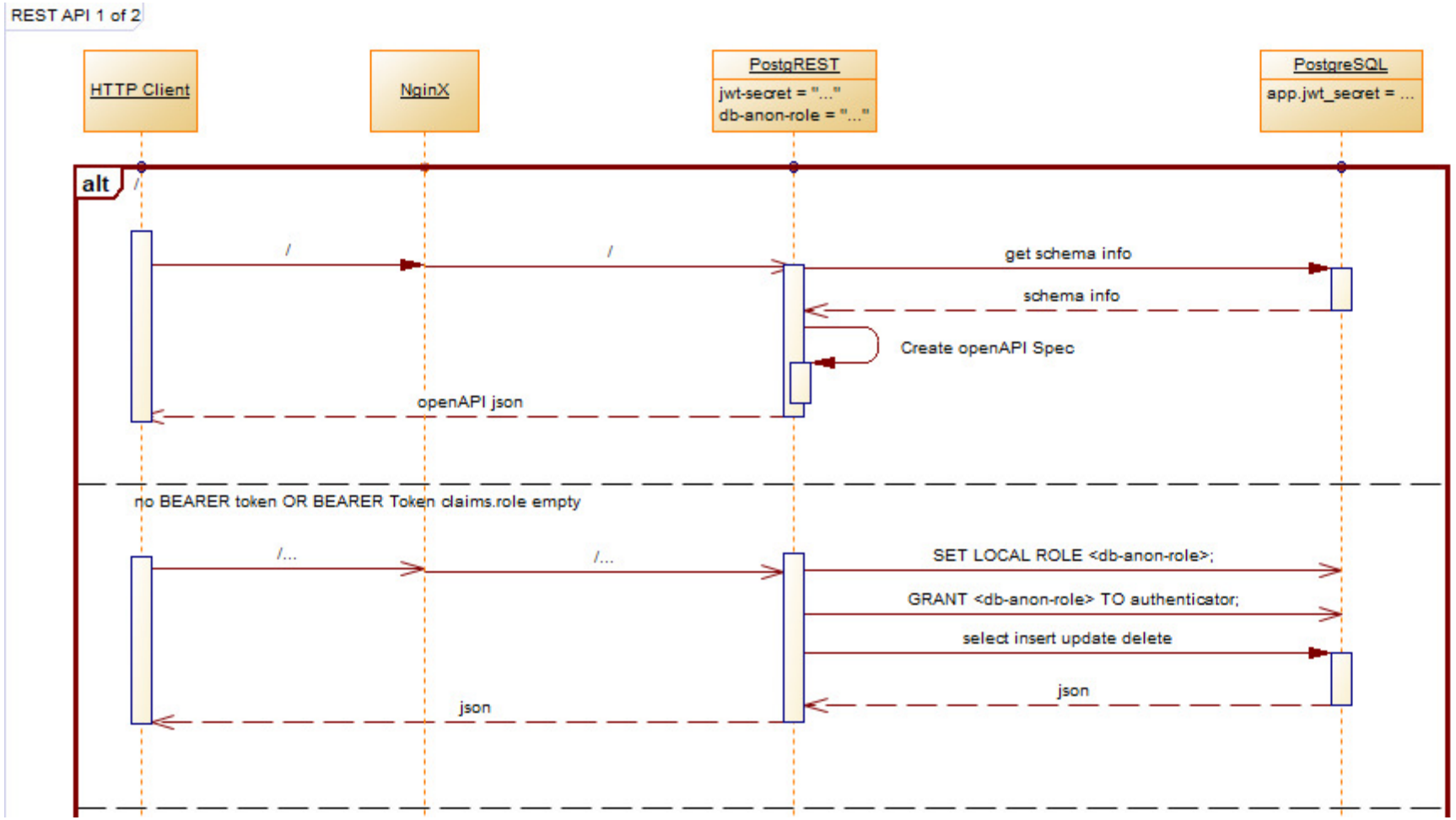
Intermezzo – Technical Architecture



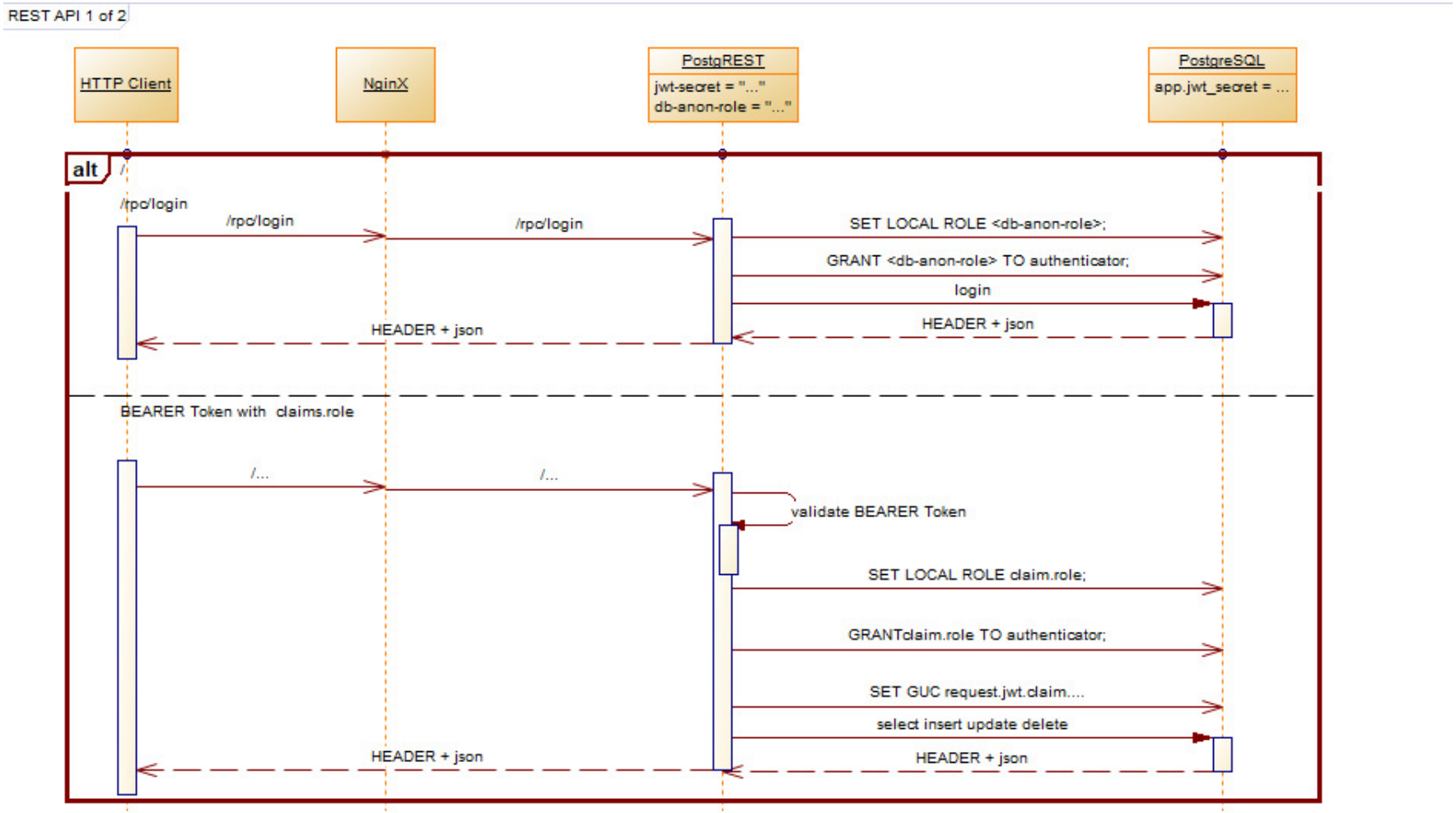
Intermezzo – Technical Architecture



REST API with minimal fuss



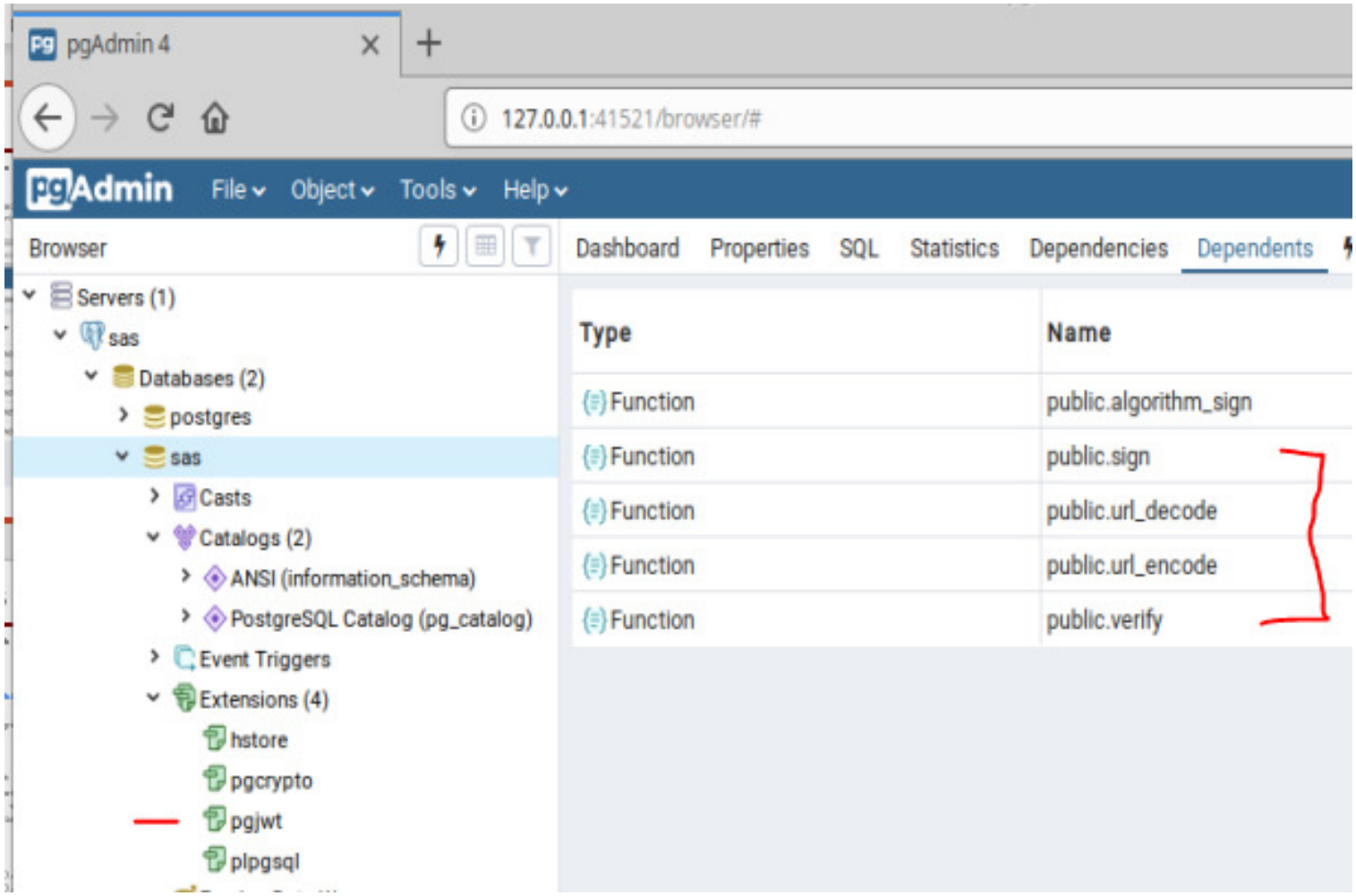
REST API with minimal fuss



JWT (‘jot’) – Extension pgjwt

```
$ sudo apt-get install git
$ sudo apt-get install postgresql-11 postgresql-server-dev-11 postgresql-contrib-11 libpq-dev


$ git clone https://github.com/michelp/pgjwt.git

$ cd pgjwt/

$ sudo make install
/bin/mkdir -p '/usr/share/postgresql/11/extension'
/bin/mkdir -p '/usr/share/postgresql/11/extension'
/usr/bin/install -c -m 644 .//pgjwt.control '/usr/share/postgresql/11/extension/'
/usr/bin/install -c -m 644 .//pgjwt--0.1.0.sql  '/usr/share/postgresql/11/extension/’

$ sudo su postgres

$ psql -d coc -c "CREATE EXTENSION pgcrypto;“

$ psql -d coc -c "CREATE EXTENSION pgjwt;"
```

JWT


pgAdmin 4
127.0.0.1:41521/browser/#
pgAdmin
File
Object
Tools
Help
Browser
Dashboard
Properties
SQL
Statistics
Dependencies
Dependents
Servers (1)
sas
Databases (2)
postgres
sas
Casts
Catalogs (2)
ANSI (information_schema)
PostgreSQL Catalog (pg_catalog)
Event Triggers
Extensions (4)
hstore
pgcrypto
pgjwt
plpgsql
Type
Name
Function
public.algorithm_sign
Function
public.sign
Function
public.url_decode
Function
public.url_encode
Function
public.verify

JWT – sign()

```
CREATE OR REPLACE FUNCTION sign( payload json
                               , secret text
                               , algorithm text DEFAULT 'HS256’
) RETURNS text
    LANGUAGE sql
AS $$
WITH
header(data) AS (
    SELECT url_encode(convert_to('{"alg":"' || algorithm || '","typ":"JWT"}', 'utf8'))
)
,payload(data) AS (
    SELECT url_encode(convert_to(payload::text, 'utf8'))
)
,signables(data) AS (
    SELECT header.data || '.' || payload.data
    FROM header, payload
)
SELECT signables.data || '.’ || algorithm_sign(signables.data, secret, algorithm)
FROM signables;
$$;
```

JWT – verify()

```
CREATE OR REPLACE FUNCTION verify( token text
                                 , secret text
                                 , algorithm text DEFAULT 'HS256’
)
RETURNS table(header json, payload json, valid boolean)
        LANGUAGE sql
AS $$
  SELECT
    convert_fromurl_decode(r[1]), 'utf8')::json AS header,
    convert_fromurl_decode(r[2]), 'utf8')::json AS payload,
    (r[3] = algorithm_sign(r[1] || '.' || r[2], secret, algorithm)) AS valid
  FROM regexp_split_to_array(token, '\.') r;
$$;
```

JWT - claims

```
DO $$
    DECLARE passphrase TEXT := 'HMAC-SHA256-passphrase_of_32+_chars_and_base64encoded_when_binary';
    DECLARE payload json := '{"role":"coc","user":"vanderwaltkarel@gmail.com"}'::json;

BEGIN
    DROP TABLE IF EXISTS _from_anonymous_block;
    CREATE TEMPORARY TABLE _from_anonymous_block AS (

    SELECT sign( payload := payload
                ,secret := passphrase
                ,algorithm := 'HS256'
               )

   ); -- TEMPORARY TABLE _from_anonymous_block
END $$;

SELECT * FROM _from_anonymous_block;
```

```
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJyb2xlIjoiY29jIiwidXNlciI6InZhbmRlcndhbHRrYXJlbEBnbWFpbC5jb20ifQ.felOHFuSHTzh
    zkw7IAmhgOi_xgWw3HbvHlw1hGiffHM"
```

JWT - header, claim payload, signature

```
DO $$
    DECLARE token TEXT := 'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9'|| -- header
                          '.eyJyb2xlIjoiY29jIiwidXNlciI6InZhbmRlcndhbHRrYXJlbEBnbWFpbC5jb20ifQ'|| -- claim payload
                          '.felOHFuSHTzhzkw7IAmhgOi_xgWw3HbvHlw1hGiffHM';  -- signature|digest|hash
    DECLARE passphrase TEXT := 'HMAC-SHA256-passphrase_of_32+_chars_and_base64encoded_when_binary';

BEGIN
    DROP TABLE IF EXISTS _from_anonymous_block;
    CREATE TEMPORARY TABLE _from_anonymous_block AS (

    SELECT *
    FROM verify( token := token
                ,secret := passphrase
                , algorithm := 'HS256'
               )

  ); -- TEMPORARY TABLE _from_anonymous_block
END $$;

SELECT * FROM _from_anonymous_block;
```

postgREST

```
$ wget https://github.com/PostgREST/postgrest/releases/download/v6.0.1/postgrest-v6.0.1-ubuntu.tar.xz

$ tar -xvf postgrest-v6.0.1-ubuntu.tar.xz

$ ./postgrest 2> try.conf
$ less try.conf

...
# Usage: postgrest FILENAME

  db-uri = "postgres://coc:coc@localhost:5432/coc"
  db-schema = “coc"
  db-anon-role = “coc"
  db-pool = 10

  server-host = "*4"
  server-port = 3000

  ## base url for swagger output
  server-proxy-uri = "http://localhost:3000/api
...

$ ./postgrest coc.conf
Attempting to connect to the database...
Listening on port 3000
Connection successful
127.0.0.1 - - [11/Aug/2019:22:24:29 +0200] "GET / HTTP/1.1" 200 - "" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0"
```

postgREST - config

Example Config File:

```
  db-uri = "postgres://user:pass@localhost:5432/dbname"
  db-schema = "public" # this schema gets added to the search_path of every request
  db-anon-role = "postgres"
  db-pool = 10
  db-pool-timeout = 10

  server-host = "!4"
  server-port = 3000

  ## unix socket location
  ## if specified it takes precedence over server-port
  # server-unix-socket = "/tmp/pgrst.sock"

  ## base url for swagger output
  # server-proxy-uri = ""

  ## choose a secret, JSON Web Key (or set) to enable JWT auth
  ## (use "@filename" to load from separate file)
  # jwt-secret = "foo"
  # secret-is-base64 = false
  # jwt-aud = "your_audience_claim"

  ## limit rows in response
  # max-rows = 1000

  ## stored proc to exec immediately after auth
  # pre-request = "stored_proc_name"

  ## jspath to the role claim key
  # role-claim-key = ".role"
```

/etc/postgrest/coc.conf

```
# Usage: postgrest FILENAME

  db-uri = "postgres://coc:coc@localhost:5432/coc"
  db-schema = "coc"
  db-anon-role = "coc"
  db-pool = 10

  server-host = "*4"
  server-port = 3000

  ## base url for swagger output
  server-proxy-uri = "http://localhost:3000/api"

  ## choose a secret to enable JWT auth
  ## (use "@filename" to load from separate file)
  #jwt-secret = "@/etc/postgrest/app.jwt_secret"
  #jwt-secret = "reallyreallyreallyreallyverysafe"
  # secret-is-base64 = false

  ## limit rows in response
  # max-rows = 1000

  ## stored proc to exec immediately after auth
  # pre-request = "stored_proc_name"
```

coc.service

```
# prefer user sans pwd over user sans shell
sudo useradd -d /home/postgrest -m postgrest -s /bin/bash

mv postgrest /home/postgrest/postgrest

$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:/
    home/kvdwalt/.dotnet/tools

#absolute paths, !NOT relative
ln -s home/postgrest/postgrest /usr/local/bin/postgrest

$ mv coc.conf /home/postgrest/.
chown postgrest:postgrest /home/postgrest/coc.conf

$ sudo mkdir /etc/postgrest
sudo chown -R postgrest:postgrest /etc/postgrest/

$ sudo ln -s /home/postgrest/coc.conf /etc/postgrest/coc.conf
```

postgREST – external passphrase

```
$ psql -c "ALTER DATABASE coc SET "app.jwt_secret" TO 'HMAC-SHA256-
    passphrase_of_32+_chars_and_base64encoded_when_binary';"

$ sudo su postgrest

$ cat << EOF > /home/postgrest/app.jwt_secret
HMAC-SHA256-passphrase_of_32+_chars_and_base64encoded_when_binary
EOF

$ ln -s /home/postgrest/app.jwt_secret /etc/postgrest/app.jwt_secret

/etc/postgrest/coc.conf
..
  ## choose a secret to enable JWT auth
  ## (use "@filename" to load from separate file)
  jwt-secret = "@/etc/postgrest/app.jwt_secret"
  #jwt-secret = "HMAC-SHA256-passphrase_of_32+_chars_and_base64encoded_when_binary"
  # secret-is-base64 = false

$ sudo journalctl -f -u coc

$ sudo systemctl restart coc
```

coc.service

```
postgrest /etc/postgrest/coc.conf

Ctrl-C

$ postgrest /etc/postgrest/coc.conf &
[1] 27196

postgrest@HP-Xubuntu:~$ ps -ef | grep postgrest
root     27147 26705  0 10:28 pts/0    00:00:00 sudo su postgrest
root     27148 27147  0 10:28 pts/0    00:00:00 su postgrest
postgre+ 27196 27149  0 10:31 pts/0    00:00:00 postgrest /etc/postgrest/coc.conf
postgre+ 27205 27149  0 10:31 pts/0    00:00:00 grep --color=auto postgrest


postgrest@HP-Xubuntu:~$ kill 27196
postgrest@HP-Xubuntu:~$
[1]+  Interrupt               postgrest /etc/postgrest/coc.conf
```

SYSTEMD – coc.service

```
sudo cp coc.service /etc/systemd/system/.

Then create the systemd service file in /etc/systemd/system/coc.service

[Unit]
Description=REST API for any Postgres database
After=postgresql.service

[Service]
ExecStart=/bin/postgrest /etc/postgrest/config
ExecReload=/bin/kill -SIGUSR1 $MAINPID

[Install]
WantedBy=multi-user.target
```

SYSTEMD

```
kvdwalt@HP-Xubuntu:~$ sudo systemctl start coc

kvdwalt@HP-Xubuntu:~$ sudo systemctl status coc
● coc.service - PostREST REST Server for PostgreSQL Database Server
   Loaded: loaded (/etc/systemd/system/coc.service; disabled; vendor preset: enabled)
   Active: active (running) since Mon 2019-08-12 12:21:18 SAST; 5s ago
     Docs: https://postgrest.com
 Main PID: 28401 (postgrest)
    Tasks: 6 (limit: 4682)
   CGroup: /system.slice/coc.service
           └─28401 /usr/local/bin/postgrest /etc/postgrest/coc.conf

Aug 12 12:21:18 HP-Xubuntu systemd[1]: Started PostREST REST Server for PostgreSQL Database Server.
Aug 12 12:21:18 HP-Xubuntu postgrest[28401]: Attempting to connect to the database...
Aug 12 12:21:18 HP-Xubuntu postgrest[28401]: Listening on port 3000
Aug 12 12:21:18 HP-Xubuntu postgrest[28401]: Connection successful

Executable path is not absolute: bash
Invalid argument `>>'

sudo journalctl --unit=coc

sudo journalctl –u coc
# tail -f
sudo journalctl -f -u coc
```

SYSTEMD

```
$ sudo systemctl enable coc
Created symlink /etc/systemd/system/postgrest.service → /etc/systemd/system/coc.service.
Created symlink /etc/systemd/system/multi-user.target.wants/coc.service →
   /etc/systemd/system/coc.service.
```

/etc/systemd/system/*.service vs /lib/systemd/system/*.service

Enacting the Lifecycle – HTTP Client e.g. curl

```
CREATE OR REPLACE FUNCTION coc.login_forever(
                username text)
RETURNS character varying
AS $BODY$
/* requires postgresql 9.6 (response.headers not returned to postgrest from 9.5 )
    SET client_min_messages TO DEBUG;
    SELECT login_forever('vanderwaltkarel@gmail.com')
*/
DECLARE header text;

DECLARE result jsonb;

BEGIN

  WITH
  .... --NEXT SLIDE
  SELECT json_build_array(json_build_object(
                  'Authorization'
                 ,FORMAT('Bearer %s', token) ))::text
  FROM _token
  INTO header ;

  RAISE DEBUG 'header: %',header;

  -- no joy
  --SET LOCAL "response.headers" TO header;
  --RAISE DEBUG 'response.headers: %', current_setting('response.headers');

  PERFORM set_config ('response.headers', header, true);
  RAISE DEBUG 'response.headers: %', current_setting('response.headers');

  WITH
  ... - NEXT SLIDE
  SELECT json_build_object( ...
                          ) INTO result
  FROM _scenario s, _option o,  _items_not_specific_to_option i ;

   --*/--*/--*/--*/

  RETURN result;

END
$BODY$;
```

exp claim

```
WITH
_payload(payload) AS (
    SELECT json_build_object( 'role', 'coc'
                              ,'user', login_forever.username::text
                              -- must be numeric; use row_to_json() or json_build_object()
                              ,'exp', (extract(epoch from now() + '5 minutes'::interval) :: integer)
                            )
 )
 ,_token(token) AS (
   SELECT sign( payload := payload
               ,secret := current_setting('app.jwt_secret')
                           )::text
   FROM _payload
 )  SELECT json_build_array(json_build_object(
                 'Authorization'
                ,FORMAT('Bearer %s', token) ))::text
 FROM _token
 INTO header ;
```

Enacting the Lifecycle – HTTP Client e.g. curl

4.1.6. "iat" (Issued At) Claim

The "iat" (issued at) claim identifies the time at which the JWT was issued. This claim can be used to determine the age of the JWT. Its value MUST be a number containing a NumericDate value. Use of this claim is OPTIONAL.

4.1.4. "exp" (Expiration Time) Claim

The "exp" (expiration time) claim identifies the expiration time on or after which the JWT MUST NOT be accepted for processing. The processing of the "exp" claim requires that the current date/time MUST be before the expiration date/time listed in the "exp" claim.

NumericDate is the last definition in Section 2. Terminology, and is defined as the number of seconds (not milliseconds) since Epoch:

Enacting the Lifecycle – HTTP Client e.g. curl

```
WITH
_scenario(id, arr) AS (
    SELECT s.id, json_agg(json_build_object( 'id', s.id, 'scenario.description', s.description, 'scenario.datecreated', s.datecreated))
    FROM scenario s
    WHERE s.username IN ('admin')
    GROUP BY s.id
) -- SELECT * FROM _scenario /*
,_option(arr) AS (
    SELECT json_agg(json_build_object( 'id', o.id, 'option.name', o.name, 'option.description', o.description ))
    FROM option o INNER JOIN scenario_option so ON so.option_id = o.id
                  INNER JOIN _scenario s ON s.id = so.scenario_id
) -- SELECT * FROM _option /*
,_qualitative_domain(id, name, arr) AS (
SELECT qd.id, qd.name,json_agg(json_build_object( 'id', dv.id
                                                 ,'position', dv.position
                                                 ,'domain_value.name', dv.name
                               )                )
FROM qualitative_domain qd INNER JOIN domain_value dv ON dv.domain_id = qd.id
GROUP BY qd.id, qd.name
) -- SELECT * FROM _qualitative_domain /*
,_items_not_specific_to_option(arr) AS (
    SELECT  json_agg(json_build_object( 'id', i.id
                                       ,'item.name', i.name
                                       ,'item.description', i.description
                                       ,'item.mayoverride', i.mayoverride
                                       ,'qualitative_domain', CASE WHEN (qd.id IS NULL) THEN NULL
                                                                   ELSE json_build_object('id', qd.id, 'domain.name', qd.name, 'domain_value', qd.arr) END
                                       ,'item.domain_value', CASE WHEN (qd.id IS NULL) THEN NULL
                                                                   ELSE json_build_object('id', dv.id, 'domain_value.name', dv.name) END
                                       ,'item.numericvalue', CASE WHEN (qd.id IS NOT NULL) THEN NULL ELSE i.numericvalue END
                                       ,'unit', CASE WHEN (qd.id IS NOT NULL) THEN NULL
                                                ELSE json_build_object('id', u.id, 'unit.description', u.description) END
                                 )      )
   FROM item i INNER JOIN scenario_item si ON si.item_id = i.id
               INNER JOIN _scenario s ON s.id = si.scenario_id
               LEFT JOIN domain_value dv ON dv.id = i.domain_value_id
               LEFT JOIN unit u ON u.id = i.unit_id
               LEFT JOIN _qualitative_domain qd ON i.qualitative_domain_id = qd.id
) -- SELECT * FROM _items_not_specific_to_option /*
SELECT json_build_object(
                          'scenario', s.arr,
                          'option', o.arr,
                          'item', i.arr
                        ) INTO result
FROM _scenario s, _option o,  _items_not_specific_to_option i ;
--*/--*/--*/--*/
```

curl

```
sudo apt-get install curl

$ curl -i -X POST "http://mentalarrow:3000/rpc/login_forever" \
-H "accept: application/json" -H "Content-Type: application/json" \
-d '{ "username": "vanderwaltkarel@gmail.com"}'

$ export TOKEN="eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9. ...  "
$ echo $TOKEN

curl "http://mentalarrow:3000/rpc/scenario_clone" -H "Content-Type: application/json" -H "Authorization: Bearer $TOKEN" \
-d "{\"description\" : \"PostgresConf South Africa 2019\", \"anc_scenario_id\" : 1, \"username\" : \"vanderwaltkarel@gmail.com\" \
,\"options\" : [{\"id\": 1, \"position\": 1},{\"id\": 2, \"position\": 2},{\"id\": 3, \"position\": 3},{\"id\": 4, \"position\": 4}] \
,\"items\" : [{\"id\": 1, \"domain_value_id\": 3},{\"id\": 2, \"numericvalue\": 50000.00},{\"id\": 3, \"numericvalue\": 60}] }"

OK !! \escape nested ""

curl "http://mentalarrow:3000/rpc/scenario_clone" -H "Content-Type: application/json"  \
-d "{\"description\" : \"PostgresConf South Africa 2019\", \"anc_scenario_id\" : 1, \"username\" : \"vanderwaltkarel@gmail.com\" \
,\"options\" : [{\"id\": 1, \"position\": 1},{\"id\": 2, \"position\": 2},{\"id\": 3, \"position\": 3},{\"id\": 4, \"position\": 4}] \
,\"items\" : [{\"id\": 1, \"domain_value_id\": 3},{\"id\": 2, \"numericvalue\": 50000.00},{\"id\": 3, \"numericvalue\": 60}] }"
```

curl

```
!! \escape nested ""

curl "http://mentalarrow:3000/rpc/scenario_delete" -H "Content-Type: application/json"  \
-d "{\"scenario_id\" : 26 }"

curl "http://mentalarrow:3000/rpc/scenario_delete" -H "Content-Type: application/json"  \
-d '{"scenario_id" : 27 }'
```

curl

```
{"message":"JWSError JWSInvalidSignature"}

{"message":"Error in $: Failed reading: satisfy"}

{"message":"JWTClaimsSetDecodeError \"Error in $.exp: expected NumericDate, encountered String\""}
```

Swagger UI - NGINX

https://github.com/swagger-api/swagger-ui

```
$ git clone https://github.com/swagger-api/swagger-ui/swagger-ui.git
```

```
Swagger UI Version | Release Date | OpenAPI Spec compatibility | Notes------------------ | --------
    ---- | -------------------------- | -----
3.13.2 | 2018-03-23 | 2.0, 3.0 | [tag v3.13.2](https://github.com/swagger-api/swagger-
    ui/tree/v3.13.2)
```

```
/home/kvdwalt/swagger-ui/dist/
```

```
/etc/nginx/sites-available
```

```
include /etc/nginx/conf.d/*.conf;
/etc/nginx/nginx.conf
```

```
user www-data;
/var/log/nginx# ls -al
```

```
/var/www/html/index.nginx-debian.html
```

```
root /var/www/coc.mentalarrow;
```

Swagger UI - NGINX

```
$ sudo su

$ su www-data

serve api on :81 say

coc.api.mentalarrow.co.za

SPA on :80

coc.mentalarrow.co.za

/var/www# mkdir coc.api.mentalarrow.co.za
/var/www# mkdir coc.mentalarrow.co.za

cp -R ./dist/* /var/www/ coc.api.mentalarrow.co.za /.

root@HP-Xubuntu:/etc/nginx/sites-available# cp /home/kvdwalt/Projects/coc/coc.api.mentalarrow.co.za .

root@HP-Xubuntu:/etc/nginx/sites-enabled# ln -s /etc/nginx/sites-available/coc.api. mentalarrow.co.za coc.api. mentalarrow.co.za
root@HP-Xubuntu:/etc/nginx/sites-enabled# ls -al
...
lrwxrwxrwx 1 root root   45 Oct  2 12:04 coc.api. mentalarrow.co.za -> /etc/nginx/sites-available/coc.api. mentalarrow.co.za
```

Swagger

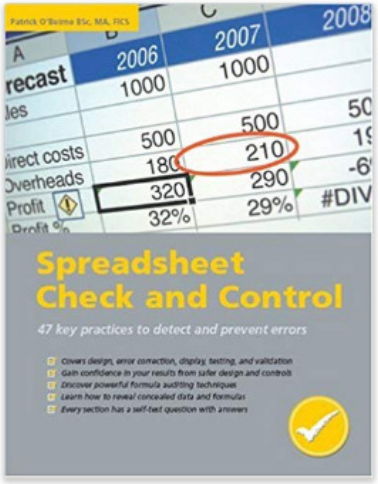
/var/www/coc.api.mentalarrow/index.html

```
<html lang="en">
<body>
...
    <script>
    window.onload = function() {
        const ui = SwaggerUIBundle({
                    //url: "https://petstore.swagger.io/v2/swagger.json",
                  //url: "http://mentalarrow:3000",
                     url: "",
                dom_id: '#swagger-ui',
                deepLinking: true,
                presets: [ SwaggerUIBundle.presets.apis, SwaggerUIStandalonePreset ],
                plugins: [ SwaggerUIBundle.plugins.DownloadUrl ],
                layout: "StandaloneLayout"
        })
      window.ui = ui
    }
    </script>
</body>
</html>
```

References

Books Advanced Search New Releases Amazon Charts Best Sellers & More The New York Times® Best Sellers Children's Books Textbooks Textbook Rentals

Books › Computers & Technology › Software




See all 2 images

Spreadsheet Check and Control Paperback – September 1, 2005

by Patrick R. O'Beirne (Author)

★★★★☆ 9 customer reviews

› See all 2 formats and editions

Paperback
$33.81

18 Used from $3.57
18 New from $24.90

Save $5.00 on orders $20.00+ 1 Applicable Promotion

May arrive after Christmas.

prime student College student? Get FREE shipping and exclusive deals LEARN MORE

This spreadsheet book is quite different from every other book on Microsoft Excel. If spreadsheet users had a driving licence, this would be their seat belt, air bag, navigation aid, repair kit, hazard indicators, and the rules of the road.
This new book describes how to produce well-crafted spreadsheets that are easy to understand, maintain, audit, and operate. It shows how to ensure data quality and accuracy and protect against

‹ Read more

Report incorrect product information.

References

Patterns of Data Modeling. Michael Blaha,
Chapter 13 Softcoded Values

Buggy spreadsheets: Russian roulette for the corporation,
The chasm between common sense and computer programming
https://www.theregister.co.uk/2006/05/03/buggy_spreadsheet/