

# SZYBKA IMPLEMENTACJA REST API

z wykorzystaniem AI





# **VOLVO GROUP DIGITAL & IT**

WHO WE ARE | OUR STRATEGIC DIRECTION | WHAT WE DO

# Volvo Group Digital & IT

Volvo Group Digital & IT manages overall digital and IT strategies and plans for the Volvo Group and has the end-to-end responsibility for all development, delivery, and support of IT solutions and services globally.



VOLVO PENTA



VOLVO ENERGY



VOLVO AUTONOMOUS  
SOLUTIONS



VOLVO FINANCIAL  
SERVICES



ARQUUS



MACK TRUCKS



RENAULT TRUCKS



VOLVO TRUCKS



VOLVO CONSTRUCTION  
EQUIPMENT



VOLVO BUSES



GROUP TRUCKS  
TECHNOLOGY



GROUP TRUCKS  
OPERATIONS



GROUP TRUCKS  
PURCHASING

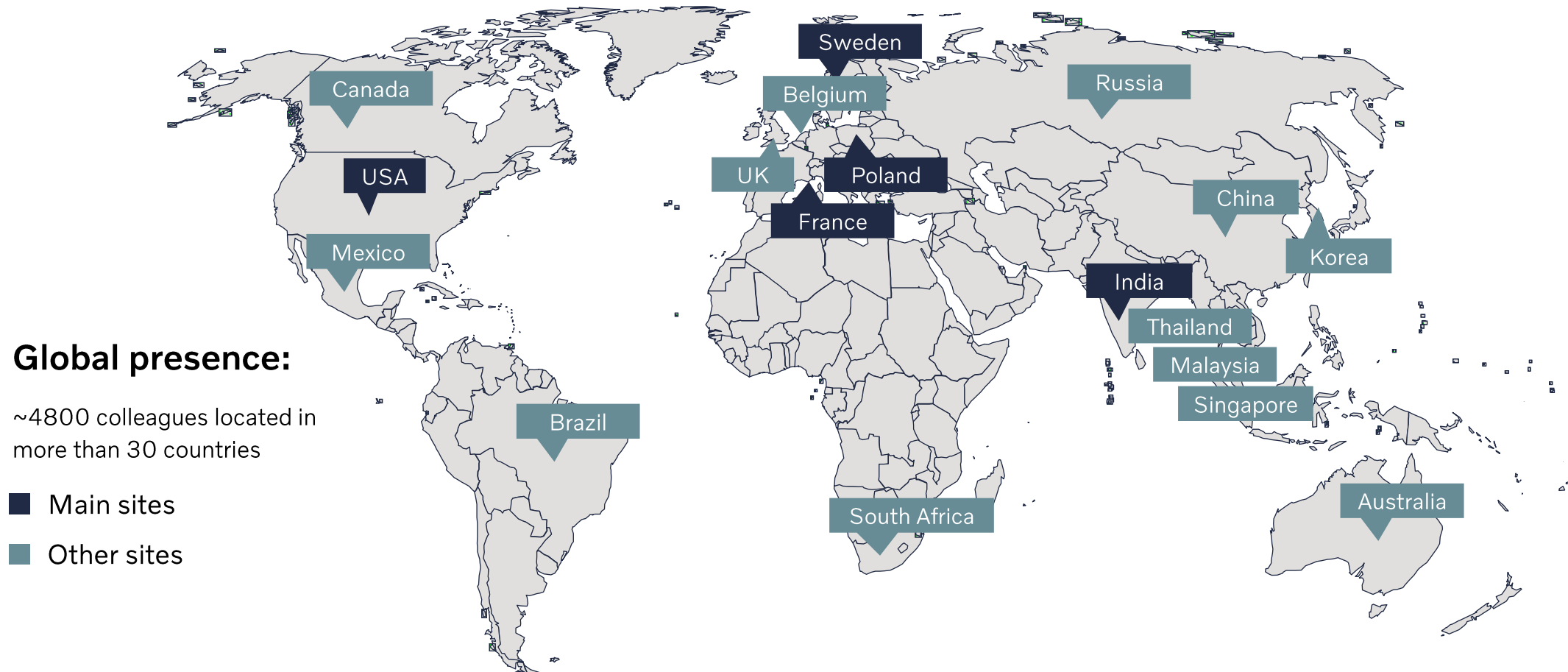


GROUP  
PEOPLE & CULTURE



GROUP  
FINANCE

# We are colleagues from around the globe



# Przygotowanie do tworzenia usług

## REST

- Zorientowanie na zasoby
- Operacje na API w nawiązaniu do metod HTTP

Zasób	POST	GET	PUT	DELETE
/customers	Utwórz nowego klienta	Pobierz wszystkich klientów	Masowa zmiana na klientach	Usuń wszystkich klientów
/customers/1	Błąd	Pobierz szczegóły dla klienta 1	Zmień szczegóły klienta 1 jeżeli istnieje	Usuń klienta 1
/customers/1/orders	Utwórz nowe zlecenie dla klienta 1	Pobierz wszystkie zamówienia dla klienta 1	Masowa zmiana zamówień dla klienta 1	Usuń wszystkie zamówienia klienta 1

- Kody stanów
  - 200(OK), 201(Created), 204(No Content), 400(Bad Request), 404(Not Found), 409(Conflict), 415 (Unsupported Media Type)
- Filtrowanie i stronicowanie
  - Parametry metody GET
  - Limit i offset

/orders?limit=25&offset=50

# Przygotowanie do tworzenia usług

## Specyfikacja OpenAPI

The OpenAPI Specification (OAS) definiuje standard, niezależnego od języka interfejsu do REST API

Specyfikacja może zostać zapisana w JSON lub YAML i definiować elementy:

- Punkty końcowe
- Parametry WE/WY dla operacji
- Metody autentykacji
- Informacje dodatkowe i kontaktowe

```
openapi: 3.0.0
info:
  title: Sample API
  description: Optional multiline or single-line description in [CommonMark](http://commonmark.org/help/) or HTML.
  version: 0.1.9

servers:
  - url: http://api.example.com/v1
    description: Optional server description, e.g. Main (production) server
  - url: http://staging-api.example.com
    description: Optional server description, e.g. Internal staging server for testing


paths:
  /users:
    get:
      summary: Returns a list of users.
      description: Optional extended description in CommonMark or HTML.
      responses:
        '200': # status code
          description: A JSON array of user names
          content:
            application/json:
              schema:
                type: array
                items:
                  type: string
```



# ILElastic

Projekt

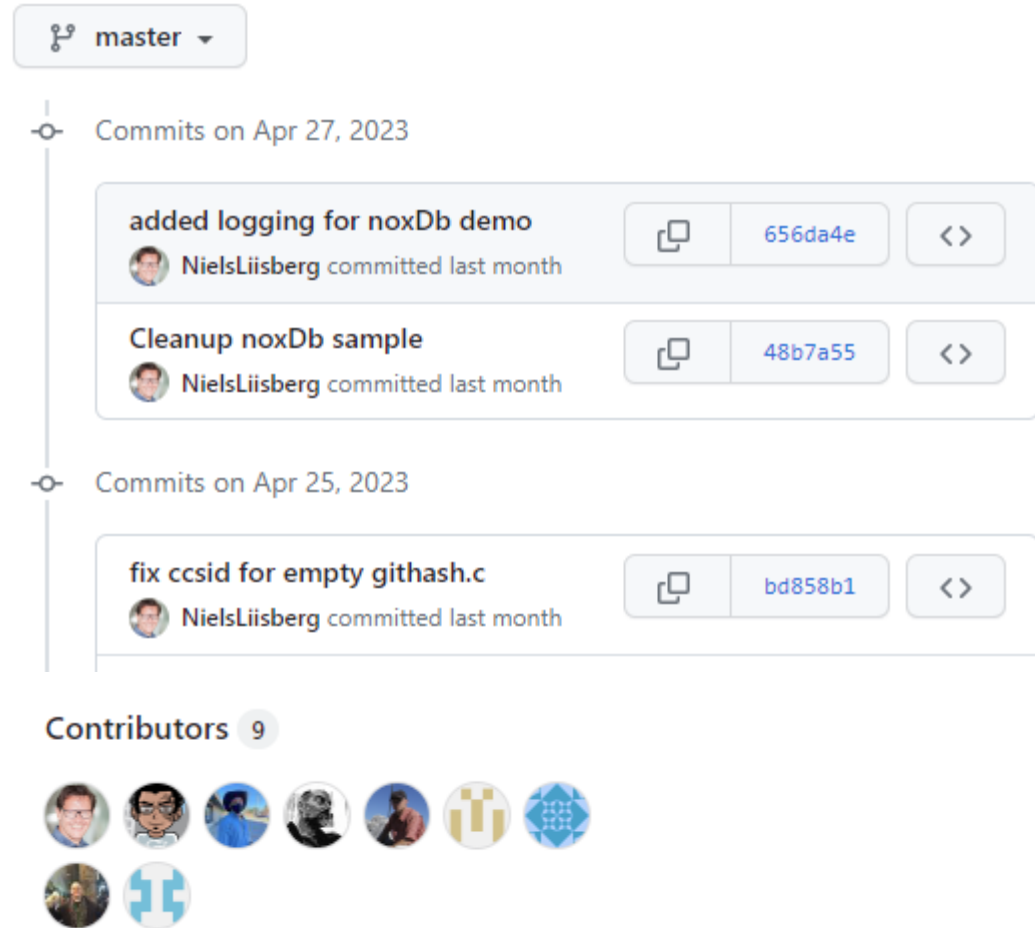
Samodzielny serwer aplikacji dla środowiska ILE na IBMi do uruchamiania mikroserwisów

 Apache-2.0 license

 41 stars

 19 watching

 25 forks



The screenshot shows the GitHub interface for the ILElastic repository. At the top, there's a branch selector set to 'master'. Below it, a timeline of commits is shown. The first section, 'Commits on Apr 27, 2023', contains two commits by NielsLiisberg: 'added logging for noxDB demo' (commit 656da4e) and 'Cleanup noxDB sample' (commit 48b7a55). The second section, 'Commits on Apr 25, 2023', contains one commit: 'fix ccid for empty githash.c' (commit bd858b1). At the bottom, the 'Contributors' section shows 9 contributors with their profile pictures.

master

Commits on Apr 27, 2023

- added logging for noxDB demo  
NielsLiisberg committed last month
- Cleanup noxDB sample  
NielsLiisberg committed last month

Commits on Apr 25, 2023

- fix ccid for empty githash.c  
NielsLiisberg committed last month

Contributors 9

# ILEastic

Szablon aplikacji

- Opcje programu oraz referencje
- Główna procedura
  - Konfiguracja
  - Dodanie ścieżek
  - Uruchomienie odbiornika
- Procedury dodatkowe
  - Struktury żądania i odpowiedzi
  - Funkcje zapisu odpowiedzi

```
ctl-opt decEdit('0,') datEdit(*YMD.) main(main);
ctl-opt debug(*yes) bndDir('ILEASTIC');
ctl-opt thread(*CONCURRENT);
/include qrppleref,ileastic

dcl-proc main;
  dcl-ds config likeds(il_config);

  config.port = 15300;
  config.host = '*ANY';

  il_addRoute(config : %paddr(helloILEASTIC) : IL_GET: '^/helloILEASTIC$');
  il_listen(config);
end-proc;

dcl-proc helloILEASTIC;
  dcl-pi *n;
    request likeds(IL_REQUEST);
    response likeds(IL_RESPONSE);
  end-pi;
  response.status = 200;
  response.contentType = 'application/json';

  il_responseWrite(response : '{"hello":"world"}');
end-proc;
```



# IBMi JSON i SQL

## Przydatne funkcje

### • JSON\_OBJECT

```
VALUES (JSON_OBJECT('first' : 'John', 'last' : 'Doe'));
```

### • JSON\_ARRAY

```
VALUES (JSON_ARRAY('Washington', 'Jefferson', 'Hamilton'));
```

### • JSON\_QUERY

```
VALUES JSON_QUERY({'id':"701", "name":{"first":"John", "last":"Doe"}}, '$.name');
```

### • JSON\_VALUE

```
VALUES (JSON_VALUE({'id':"987"}, '$.id' RETURNING INTEGER));
```

### • JSON\_TABLE

```
{'id':903, 'name' : { 'first':"Mary", 'last':"Jones" }, 'office' : "E-739"}
```

```
SELECT t.first, t.last, t.office
FROM emp,
     JSON_TABLE(
         emp.jsondoc,
         'lax $'
         COLUMNS (
             first VARCHAR(10) PATH 'lax $.name.first',
             last VARCHAR(10) PATH 'lax $.name.last',
             office VARCHAR(10) PATH 'lax $.office'
         )
     ) AS t;
```

```
select json_object ('id' value id,
                    'name' value json_object ( 'first' value first_name,
                                                'last' value last_name),
                    'office' value office_number
                    absent on null)
from empdata;
```

```
{'id':901,'name':{'first':"John",'last':"Doe"},"office":"E-334"}
```

# GitHub Copilot

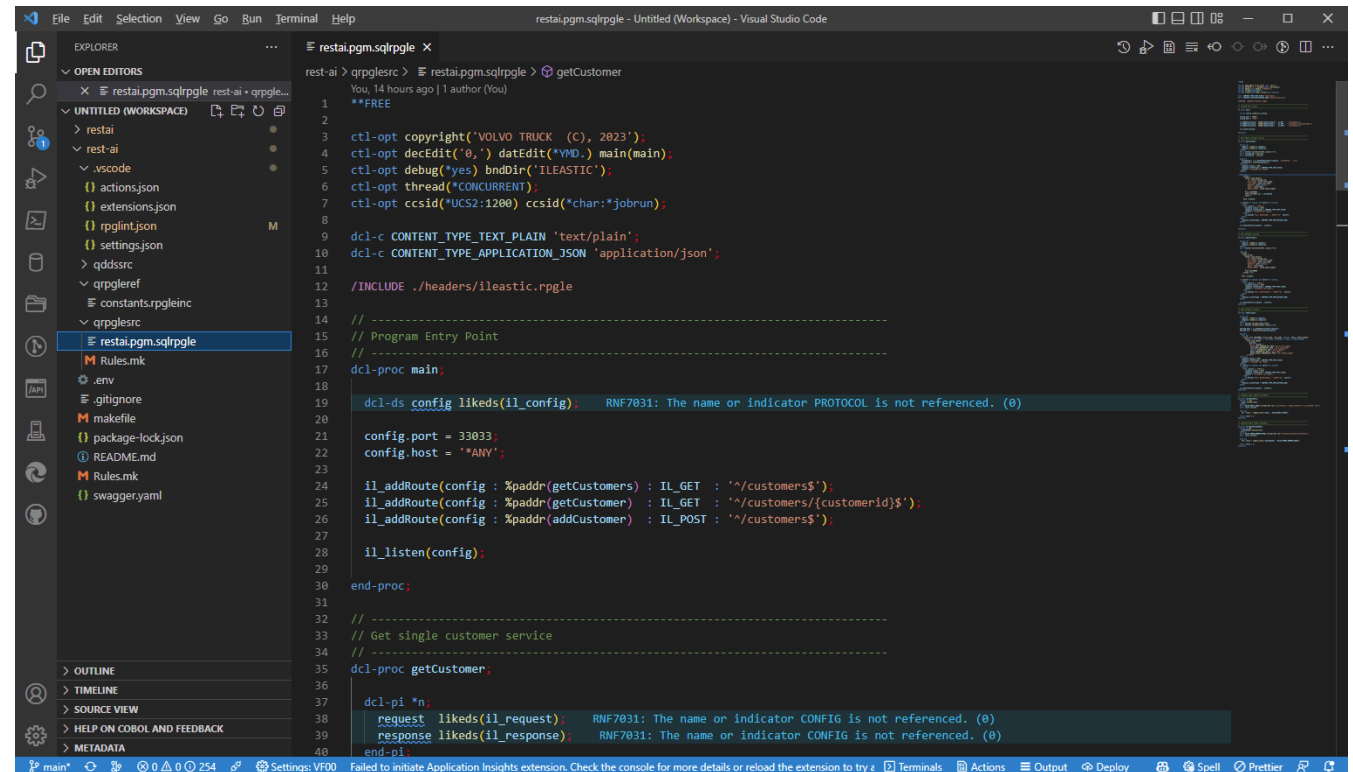
Your AI pair programmer

- Narzędzie oparte na sztucznej inteligencji
- Analizuje ogromną ilość publicznie dostępnych kodów źródłowych
- Generuje inteligentne sugestie i podpowiedzi kodu w czasie rzeczywistym.
- Obsługuje wiele języków programowania, takich jak Python, JavaScript, Ruby, C++ i wiele innych.
- Ułatwia pisanie kodu poprzez automatyczne generowanie fragmentów kodu.
- Uczy się na bieżąco dzięki interakcji z użytkownikami, co przyczynia się do poprawy jakości generowanego kodu.

# Code for IBM i

IBM i development extension for VS Code

- Podpowiadanie kodu
- Formatowanie
- Akcje
- Deployment
- Uruchamianie
- Debugowanie



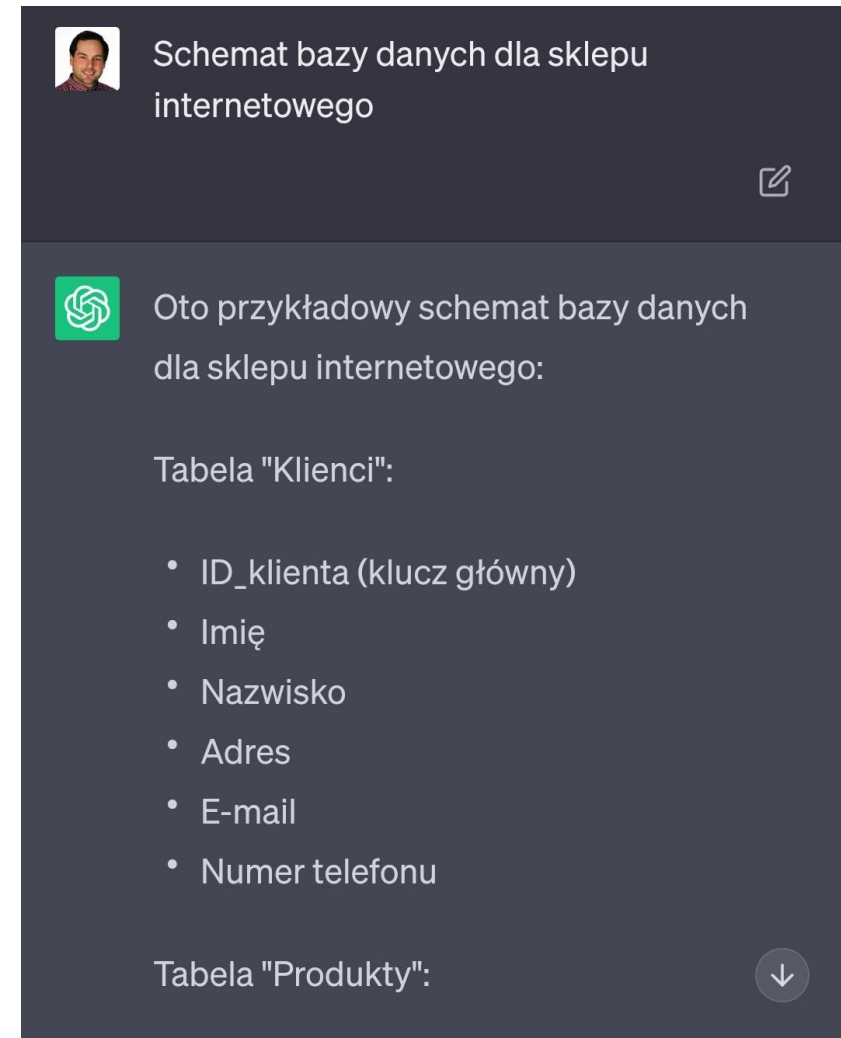
```

restai.pgm.sqlrpgle
rest-ai > qrpqlsrc > restai.pgm.sqlrpgle > getCustomer
You, 14 hours ago | 1 author (You)
**FREE
1
2
3   ctl-opt copyright('VOLVO TRUCK (C), 2023');
4   ctl-opt decEdit('0,') datEdit('YMD.') main(main);
5   ctl-opt debug(*yes) bndDir('ILEASTIC');
6   ctl-opt thread(*CONCURRENT);
7   ctl-opt ccsid(*UCS2:1200) ccsid(*char:*jobrun);
8
9   dcl-c CONTENT_TYPE_TEXT_PLAIN 'text/plain';
10  dcl-c CONTENT_TYPE_APPLICATION_JSON 'application/json';
11
12  /INCLUDE ./headers/ileastic.rpgle
13
14  // -----
15  // Program Entry Point
16  // -----
17  dcl-proc main;
18
19  dcl-ds config likeds(il_config);  RNF7031: The name or indicator PROTOCOL is not referenced. (0)
20
21  config.port = 33033;
22  config.host = '*ANY';
23
24  il_addRoute(config : %paddr(getCustomers) : IL_GET : '^/customers$');
25  il_addRoute(config : %paddr(getCustomer) : IL_GET : '^/customers/{customerid}$');
26  il_addRoute(config : %paddr(addCustomer) : IL_POST : '^/customers$');
27
28  il_listen(config);
29
30 end-proc;
31
32 // -----
33 // Get single customer service
34 // -----
35 dcl-proc getCustomer;
36
37 dcl-pi *n;
38   request likeds(il_request);  RNF7031: The name or indicator CONFIG is not referenced. (0)
39   response likeds(il_response);  RNF7031: The name or indicator CONFIG is not referenced. (0)
40 end-pi;

```

# Aplikacja

- Pomysł na strukturę aplikacji
- Dokumentowanie
- Uszczegóławianie



# Baza danych

- AI nie czyta w myślach
- Komentarze
- Kontynuowanie kodu
- Wybór propozycji

```
-- Set label on column fields with texts
LABEL ON Customers IS 'Customers';
```

```
-- Set label on column fields with texts
LABEL ON COLUMN Customers.customer_id TEXT IS 'Customer ID';
```

```
rest-ai > qddsrc > ☰ customers.table
1  -- SQL for creating the database
2  -- +-----+
3  -- | Customers |
4  -- +-----+
5  -- | - customer_id (PK) |
6  -- | - first_name |
7  -- | - last_name |
8  -- | - email |
9  -- | - password |
10 -- | - address |
11 -- | - phone_number |
12 -- +-----+
13
14 CREATE TABLE Customers (
    customer_id INT NOT NULL AUTO_INCREMENT,
    first_name VARCHAR(255) NOT NULL,
    last_name VARCHAR(255) NOT NULL,
    email VARCHAR(255) NOT NULL,
    password VARCHAR(255) NOT NULL,
    address VARCHAR(255) NOT NULL,
    phone_number VARCHAR(255) NOT NULL,
    PRIMARY KEY (customer_id)
);
```

# Baza danych

Dodanie informacji o używanej bazie danych – IBMi DB2 powoduje dostosowanie sugestii

```
qddsrc > ≡ products.table
-- SQL for creating customers table on IBMi DB2
-- +-----+
-- | Products      |
-- +-----+
-- | - product_id (PK)|
-- | - name          |
-- | - description    |
-- | - price          |
-- | - stock          |
-- +-----+

CREATE TABLE products (
    product_id INT NOT NULL GENERATED ALWAYS AS IDENTITY (START WITH 1, INCREMEN
    name VARCHAR(50) NOT NULL,
    description VARCHAR(255) NOT NULL,
    price DECIMAL(10,2) NOT NULL,
    stock INT NOT NULL,
    PRIMARY KEY (product_id)
);
```

# Procedury

- Jakość idzie w parze z ilością
- Dane testowe
- Pułapki
- Korzyści

```
// -----
// Validate email address procedure
// -----
dcl-proc validateEmail;
  dcl-pi *n ;
    email varchar(255);
  end-pi;

  dcl-s validEmailRegex varchar(255) inz['^[a-zA-Z0-9_.-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-]+$'];
```

```
211 // -----
212 // Get products service
213 // -----
214 // -----
215 dcl-proc getProducts;      You, now * Uncommitted changes

  dcl-pi *n;
    request Likeds(il_request);
    response Likeds(il_response);
  end-pi;
  dcl-s content varchar(32740) ccsid(*utf8);

  exec sql
    values (
      json_array(
        select json_object(
          'id' value product_id,
          'name' value name,
          'description' value description,
          'price' value price
        )
        from PRODUCTS
        format json)
      )
    into :content;

  if (sqlstt <> *blanks and sqlstt <> *zeros);
    select;
      when (sqlstt = '22002');
        response.status = 400;
        response.contentType = CONTENT_TYPE_TEXT_PLAIN;
        content = 'Products not found';
      other;
        il_joblog('Proc: getProducts :: SQLSTT:%s': sqlstt);
    ends;
  else;
    response.contentType = CONTENT_TYPE_APPLICATION_JSON;
  endif;
```



# Dodatki

- Makefile
- Rules.mk
- Swagger.yaml

```

openapi: '3.0.2'    "openapi": Unknown word.
info:
  title: API Title
  version: '1.0'
servers:
  - url: https://api.server.test/v1
paths:
  # Add get customers path
  /customers:
    get:
      Try it
      responses:
        '200':
          description: OK
  # Add get customer path
  /customers/{id}:
    get:
      responses:
        '200':
          description: OK

```

```

# Get customer path
/customers/{id}:
  get:
    Try it
    parameters:
      - name: id
        in: path
        required: true
        schema:
          type: integer
    responses:
      '200':
        description: OK

```

```

rest-ai > qddsrc > M Rules.mk
1  DEPARTMENT.FILE: department.table
2  EMPLOYEE.FILE: employee.table
3  CUSTOMER.FILE: customer.table
4  PRODUCT.FILE: product.table
5

```

V O L V O

THANK YOU