**VerivoxTariffComparisor Web API**

# Summary

VerivoxTariffComparisor is a web APIservice which is mainly designed to calculate the Annual expenditure of Electricity cost from different tariif provider with their feature. This web API contains 3 different services

1. VerivoxTarifComparisor- returns the Annual cost results between tariffs through REST WebAPI. TariffComparisor reads the different TariffProducts from tarrifProvider WEB API Service.
2. TarifProvider- Is a REST API Services with CRUD operations for Product i.e. tariffProvider .However as per requirement there are 2 products Product A and Product B is already available by default. Any number of Tariff Product can be added
3. ProductType- A REST API service with CRUD operations for Product Type ( A feature in Prudct) model

Tech used

* C# programming Language
* .Net Core 3.1
* .Net 8.0 framework (SDK)
* EF Core for RDMS
* InMemory Database for TariffProvider Products
* MSUnitTest

# Installation

1. Download the files from GitHub “https://github.com/Punitha854/Verivox” into any folder both in linux or windows.
2. Navigate to the downloaded folder explorer “...\VerivoxTariffComparisor”
3. Open cmd window as administrator and run command

*docker build -t tariffcomparisor . && docker run -p 8080:8080 tariffcomparisor*

*Note : Make sure your docker desktop is installed. Do not remove the period in docker command*

*A screenshot of a computer program

Description automatically generated*

Now the TariffComparisor services is already running. Open docker desktop and make sure image and container is running

A screenshot of a computer

Description automatically generated

Open the browser window and execute URL

# Running the Services

## SWAGGER

<http://localhost:8080/swagger/index.html>

A screenshot of a computer

Description automatically generated

## Browser

<http://localhost:8080/CompareProductAProductBTariffs?consumption=4500>

[{"tariffName":"Packaged Tariff","annualCost":950,"message":"AnnualCost calculated sucessfully"},{"tariffName":"Basic Electricity Tariff","annualCost":1050,"message":"AnnualCost calculated sucessfully"}]

## Postman

curl -X 'GET' \

'http://localhost:8080/CompareProductAProductBTariffs?consumption=4500' \

-H 'accept: text/plain'

## Visual Studio:

To Open the software in Visual studio, go to the downloaded path “...\VerivoxTariffComparisor” and then open the .sln file. The solution contains both service project and test project as well

A screenshot of a computer

Description automatically generated

# Other services:

## VerivoxTariffCompariosr

Tariff comparisor has 3 Web API which yield calculations

1 .Return Calculation result between Product A and Product B Example: *http://localhost:8080/CompareProductAProductBTariffs?consumption=4500*

2. Return Calculation result for all the Tariff product selected by their comma separated names by user Example: *http://localhost:8080/CompareGivenProductTariffs?tariffProducts=Product%20A%2CProduct%20B%2CProduct%20C&consumption=3500*

3 Return Calculation result for all the Tariff products in database:Example *http://localhost:8080/CompareAllProductTariffs?consumption=4000*

## Verivox Tariff Provider:

Tariff Provider is REST API service with CRUD operation to read write and update tariff provider products. By default , 2 products Product A and Product B already exists in the service

Post Man example to input new product

{

"name": "Product C",

"type": 3

"productType": {

"id": 3,

"typeName": "Special Tariff"

},

"unitKwhCostInEuro": 0.25,

"baseCostPerMonth": 0,

"baseCostPerYear": 700,

"addtionalkwhCost": 0,

"includedKwhCost":3000}

curl -X 'POST' \

'http://localhost:8080/api/TariffProvider' \

-H 'accept: text/plain' \

-H 'Content-Type: application/json' \

-d '{

"name": "Product C",

"type": 3,

"productType": {

"id": 3,

"typeName": "Special Tariff"

},

"unitKwhCostInEuro": 0.25,

"baseCostPerMonth": 0,

"baseCostPerYear": 700,

"addtionalkwhCost": 0,

"includedKwhCost":3000

}'

A screenshot of a computer

Description automatically generated

## VerivoxTariffComparisor.Test

This is a test which has to be executed in CI/CD pipelines

Task

**Task: Tariff Comparison**

**Background**

Suppose you are working on a platform to compare Electricity prices, where users can estimate their annual cost based on consumption. Assume there is an external provider of Electricity Tariffs. Your job is to process the user input and do the appropriate calculations depending on the tariff type.

This is an extract of the items returned by the *Tariff Provider*:

[

{"name": "Product A", "type": 1, "baseCost": 5, "additionalKwhCost": 22},

{"name": "Product B", "type": 2, "includedKwh": 4000, "baseCost": 800, "additionalKwhCost": 30},

...

]

The type of product determines the calculation model as this

1. Product A • Consumption: 3500 kWh/year => Annual costs = 830 €/year (5€ \* 12 months = 60 € base costs + 3500 kWh/year \* 22 cent/kWh = 770 € consumption costs)

• Consumption: 4500 kWh/year => Annual costs = 1050 €/year (5€ \* 12 months = 60 € base costs + 4500 kWh/year \* 22 cent/kWh = 990 € consumption costs)

• Consumption: 3500 kWh/year => Annual costs = 800 €/year

• Consumption: 4500 kWh/year => Annual costs = 950 €/year (800€ + 500 kWh \* 30 cent/kWh = 150 € additional consumption costs)

Type: “1 - basic electricity tariff”

Calculation model: base costs per month 5 € + consumption costs 22 cent/kWh. Examples:

2. Product B

Type: “2 - Packaged tariff”

Calculation model: 800 € for up to 4000 kWh/year and above 4000 kWh/year additionally 30 cent/kWh. Examples:

**Task**

Create a service to read the products from the *Tariff Provider* (you can mock it as you consider better), do the calculations, and return the results, considering the following aspects:

* + • Develop a model to build up the two Products mentioned above, and to compare these products based on their annual costs. The comparison should accept the following input parameter: o Consumption (kWh/year)
  + • Return a list of the calculation results with at least the columns: o Tariff name
  + o Annual costs (€/year)
  + • The list should be sorted by costs in ascending order
  + • Create a RESTful service to retrieve the results
  + • Consider there may be more products and product types
  + • Bonus point: set up a script to quickly install all application requirements and run it on team member computers if they are running Linux
  + • Note: Please implement this task in C#, NodeJS or PHP