Kyklo – code exercise

Requirements

Rails 4.x application which expose public interface

Public API Interface responds to the following requests:

#1 [GET] models/:model_slug/model_types

#2 [POST] models/:model_slug/model_types_price/:model_type_slug

Authentication method can be chosen by you. If necessaryAPI needs to be secured as server is exposed to the public.

Required Models:

Organization
  domain attributes:
    :name, :public_name, :type, :pricing_policy
    Type can be "Show room", "Service", "Dealer"
    Pricing Policy described in the section 'pricing logic'

Model (refers to the Car Model)
  is linked with an Organization
  domain attributes:
   :name, :model_slug,

ModelType
  is linked with an Model
  domain attributes:
   :name, :model_type_slug, :model_type_code, :base_price

Requests:

#1 GET models/:model_slug/model_types
returns list of models types for given model, for each of the model types
returns 'total_price' calculated based on the 'pricing_policy' logic
format of example output


```
{
  "models": [
    {
      "name": "serie_1",
      "model_types": [
        {
          "name": "bmw_116i",
          "total_price": 180000
        },
        {
          "name": "bmw_125i",
          "total_price": 205000
        }
      ]
    },
```


#2 [POST] models/:model_slug/model_types_price/:model_type_slug
parameters:
base_price: Integer
model_slug: 'serie_3'
model_type_slug: '330i'

Returns model_type's data, including calculated total_price based on the organization 'pricing policy' for the given model_type considering base_price passed as a POST parameter.

format of example output:


```
{
  "model_type": {
    "name": "330i",
    "base_price": 200000,
    "total_price": 240000
  }
}
```


Pricing policy - logic

Based on the Organization pricing policy which can be set as:
"Flexible", "Fixed", "Prestige"

Logic for calculating model type price

where margin is a dynamic factor and depends on the type of the pricing policy:
for the different types of policies can be describe in the following way:

"Flexible":
margin is equal, how many letters 'a' can you find on this site http://reuters.com
divided by 100
total_price = base_price * margin

"Fixed"
margin is equal, how many words 'status' can you find on site
https://developer.github.com/v3/#http-redirects
total_price = base_price + margin

"Prestige"
margin is equal, how many pubDate elements can you find on page
http://www.yourlocalguardian.co.uk/sport/rugby/rss/
total_price = base_price + margin

While building this application:
* use git and commit possible often using descriptive commits descriptions
* use OOP principles
* use TDD and any test framework which is comfortable for you to build this application
* use any gem and data storage which you can find useful