Introduction of API-only Rails app

Function

This is a gas price reporting application and it has several features

1. Users can view all gas stations and their gas prices.
2. Users can create/update/delete/ gas prices.
3. Users can get the cheapest nearby gas prices.

Implementation

Set Controllers

Cause this is a sample application, after create a new Rails application I generate a `scaffold` for `Gasstation` model.

```
$ rails generate scaffold Gasstation prices:float latitudes:float
longitudes:float
```

And then I generate a `base controller` and `gasstation controller` in the path `app/views/api/v1`

```
$ rails g controller api/v1/base --no-assets
```

```
$ rails g controller api/v1/gasstations --no-assets
```

So I will implement the functions as actions mainly in this `gasstation controller`

view prices

First, users can view all the gasstations or a certain gas station.

For view all stations, the HTTP method should be `GET` and it will process in `index` action and the `Gasstation` model will retrieve all the stations. After that, render them as JSON.

For view a certain station, the HTTP method should be `GET` and it will process in `show` action and the `Gasstation` model will retrieve a record with a certain `gasstation_id`. For example, the user touch a gas station on his phone and the gas station's id will be send as a URL parameter to the server.

app/controllers/api/v1/gasstations_controller.rb

```
# GET /gasstations
def index
  @gasstations = Gasstation.all
  render json: @gasstations
end

# GET /gasstations/1
def show
  @gasstation = Gasstation.find(params[:id])
  render json: @gasstation
end
```

create/updata/delete prices

Second, for `RESTful API`, I need to implement `create/updata/delete` actions in the controller.

For `create` , the method should be `POST` and the `Gasstation` model will create a new record accroding to the parameters(prices, lat, lng). And then save the record and render the page.

For `update` , the method should be `PATCH/PUT` and the `Gasstation` model will retrieve a certain record accroding to the parameters(station id) and update that.

For `destroy`, the method should be `DELETE`.

app/controllers/api/v1/gasstations_controller.rb

```
# POST /gasstations
def create
  @gasstation = Gasstation.new(gasstation_params)

  if @gasstation.save
    render json: @gasstation, status: :created, location:
api_v1_gasstation_url(@gasstation)
  else
    render json: @gasstation.errors, status: :unprocessable_entity
  end
end
```

```ruby
# PATCH/PUT /gasstations/1
def update
  @gasstation = Gasstation.find(params[:id])
  if @gasstation.update(gasstation_params)
    render json: @gasstation
  else
    render json: @gasstation.errors, status: :unprocessable_entity
  end
end

# DELETE /gasstations/1
def destroy
  @gasstation = Gasstation.find(params[:id])
  @gasstation.destroy
end
```

Get the cheapest nearby gas prices

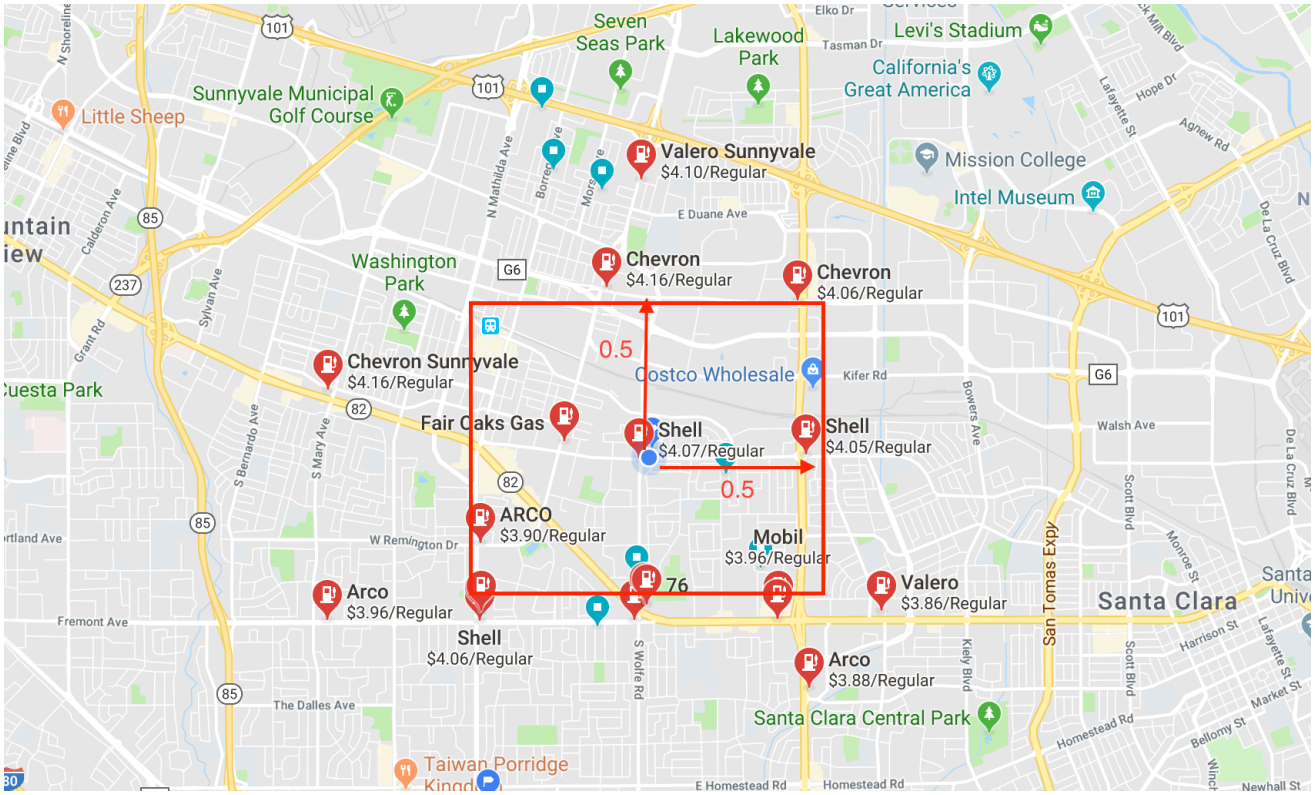
For get the cheapest nearby gas prices, my thought are as follows.

If this feature is present in a mobile app, then we need to display a map and then mark the gas stations near the current location. It also displays a sorted list that sorts the gas stations on the map from the prices low to high. This way, the user can view which gas station is the cheapest.

Therefore, it is not necessary to calculate the distance, but to find the lowest gas price in a certain range. If it doesn't find a gas station, recursively call the function until it find a gas station.

So I centered on the current user location and draw a square on the map using an `offset` parameter(This can be defined by the user). Only the gas stations with longitude and latitude in this area are what we need, and record the lowest gas price and the corresponding gas station id.

If it cannot find any station, then call this function again and `offset = offset + 0.5`



app/controllers/api/v1/gasstations_controller.rb

```
# GET /gasstations/cheapest
def get_cheapest(offset=0.5)

  cur_lat = params[:lat].to_f
  cur_lng = params[:lng].to_f
  lowest_price = 10
  station_id = 0

  gasstations = Gasstation.all

  gasstations.each do |gasstation|
    if  cur_lat - offset <= gasstation.latitudes and gasstation.latitudes <=
cur_lat + offset and cur_lng - offset <= gasstation.longitudes and
gasstation.longitudes <= cur_lng + offset
      p gasstation
      if gasstation.prices < lowest_price
        lowest_price = gasstation.prices
        station_id = gasstation.id
      end
    end
  end

  if lowest_price == 10
    get_cheapest(offset + 0.5)
```

```
  else
    @gasstation = Gasstation.find(station_id)
    render json: @gasstation
  end
end
```

Seed Data

I create some seed data for testing.

db/seeds.rb

```
20.times do |n|
  prices  = format("%.2f", rand(1.95...2.95))
  latitudes = format("%.2f", rand(35.5...38.5))
  longitudes = format("%.2f", rand(120.0...124.0)*(-1))

  Gasstation.create!(prices: prices, latitudes: latitudes, longitudes:
longitudes)
end
```

Routes

config/routes.rb

```
namespace :api do
    namespace :v1 do
      resources :gasstations do
        collection do
          get 'cheapest', to: 'gasstations#get_cheapest'
        end
      end
    end
  end
```

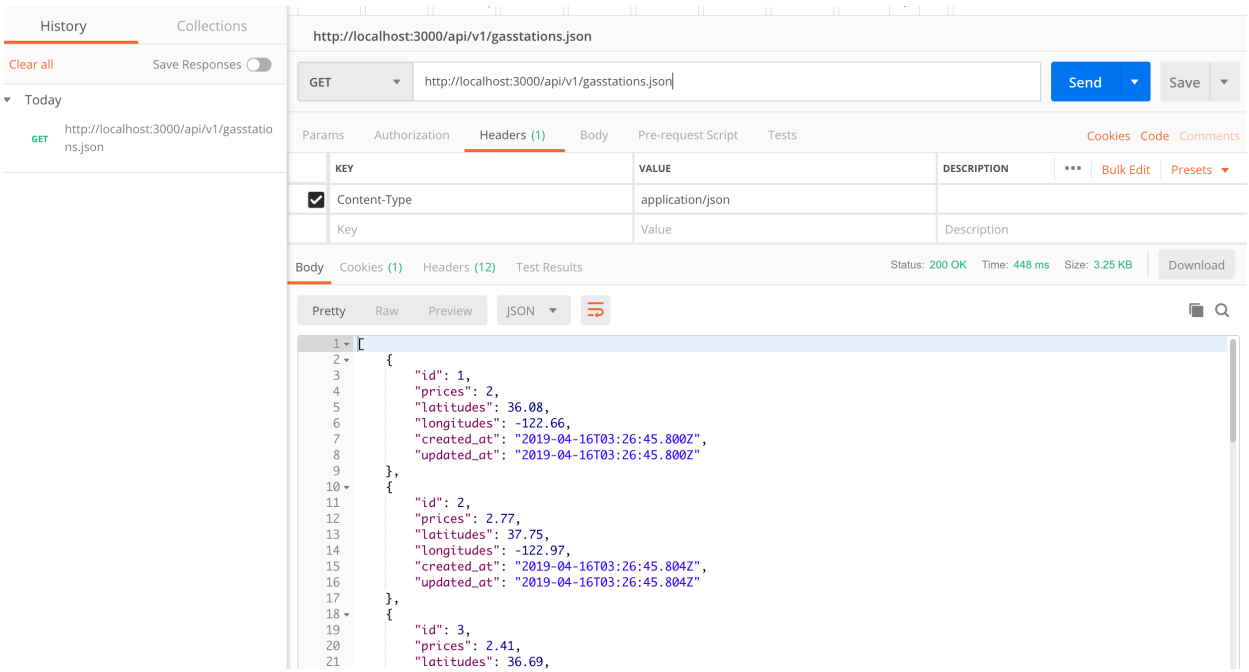
All routes:

```
cheapest_api_v1_gasstations
  GET    /api/v1/gasstations/cheapest(.:format)
api/v1/gasstations#get_cheapest

api_v1_gasstations
  GET    /api/v1/gasstations(.:format)
api/v1/gasstations#index
  POST   /api/v1/gasstations(.:format)
api/v1/gasstations#create

api_v1_gasstation
  GET    /api/v1/gasstations/:id(.:format)
api/v1/gasstations#show
  PATCH  /api/v1/gasstations/:id(.:format)
api/v1/gasstations#update
  PUT    /api/v1/gasstations/:id(.:format)
api/v1/gasstations#update
  DELETE /api/v1/gasstations/:id(.:format)
api/v1/gasstations#destroy
```

Test with Postman

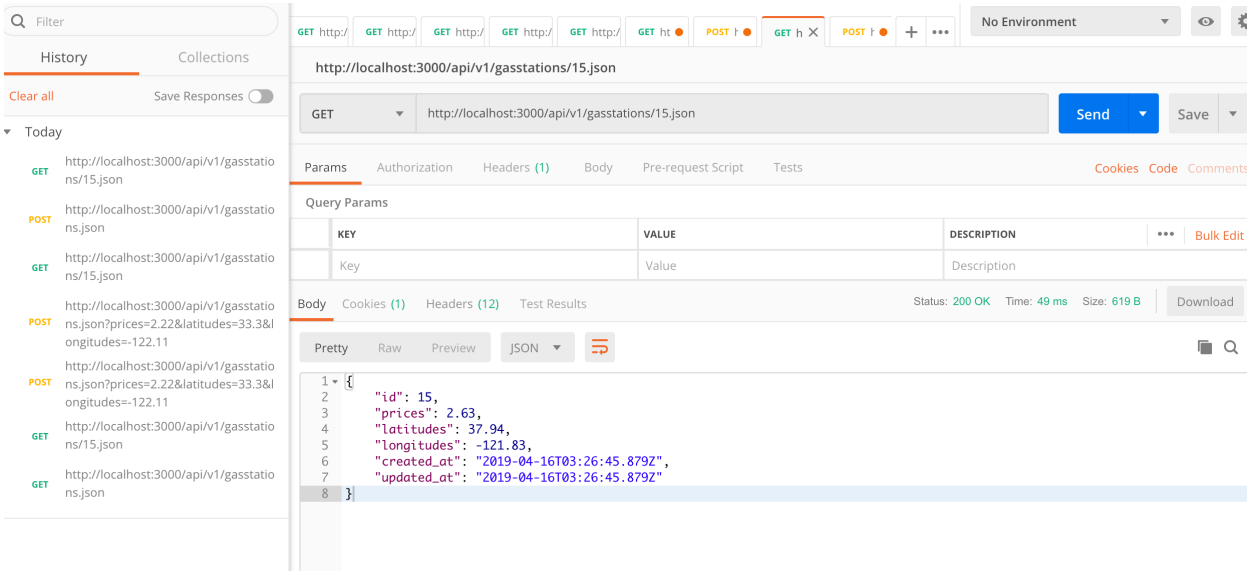
1. View all gas stations

```
http://localhost:3000/api/v1/gasstations.json
```



2. View a certain station

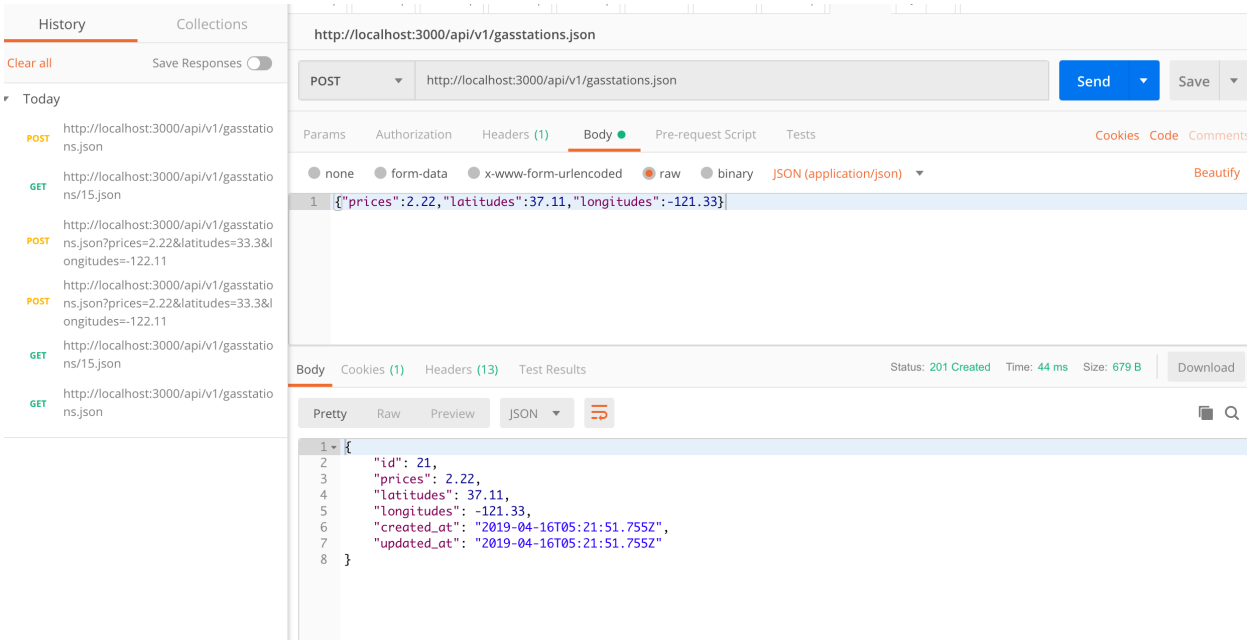
```
http://localhost:3000/api/v1/gasstations/15.json
```



3. Create/Updata/Delete

```
http://localhost:3000/api/v1/gasstations.json
```

```
{
    "id": 15,
    "prices": 2.63,
    "latitudes": 37.94,
    "longitudes": -121.83,
}
```



For update, we need to give it an station id and a set of json data.

For delete, we need to give it an station id.

```
http://localhost:3000/api/v1/gasstations/15.json
```

4. Get cheapest

   Input the location as parameters

```
http://localhost:3000/api/v1/gasstations/cheapest.json?lat=33&lng=-122
```