TODO phase1

1. 要件定義に記載された各エンドポイントに対するアクセスについてのテスト
   を.tests/Feature/ProductTest.php関数に実装する.テスト項目は以下の通り.また各テスト用関数は,ステータスコード200のHTTPレスポンスを返却する.
   - api/productsにGETメソッドでアクセスできる
     canAccessApiProductsByGET()
   - api/productsにPOSTメソッドでアクセスできる
     canAccessApiProductsByPOST()
   - api/products/{product_id}にGETメソッドでアクセスできる
     canAccessApiProductsProduct_idByGET()
   - api/products/{product_id}にPUTメソッドでアクセスできる
     canAccessApiProductsProduct_idByPUT()
   - api/products/{product_id}にDELETEメソッドでアクセスできる
     canAccessApiProductsProduct_idByDELETE()
   - api/shopsにGETメソッドでアクセスできる
     canAccessApiShopsByGET()
   - api/shopsにPOSTメソッドでアクセスできる
     canAccessApiShopsWithByPOST()
   - api/shops/{shop_id}にGETメソッドでアクセスできる
     canAccessApiShopsShop_idByGET()
   - api/shops/{shop_id}にPUTメソッドでアクセスできる
     canAccessApiShopsShop_idByPUT()
   - api/shops/{shop_id}にDELETEメソッドでアクセスできる
     canAccessApiShopsShop_idByDELETE()

   finished all at 7/24 13:55
2. 上記関数に対応するように,routes/api.phpに各エンドポイント に対する処理を記述する.まず簡単にclosureで記述する.関数に実
   際の処理は記述しない.
   finished all at 7/24 14:05
3. テスト用にデータベースを作成する.
   - テスト用データベースを２種類作成(Products, shops)
   - 要件通り各カラムを設定
   - fakerを利用してseederを作成.テスト用seederの要件は以下の通り.
     - productは10品目,shopは5店舗存在する.
       finished all at 7/24 16:30
4. データベースに関するテストを実装して検証する.検証する要件は以下の通り.

- api/productsにGETメソッドでアクセスするとJSONで商品情報が10件返却される.より粒度を細かくすると以下の通り.
  - api/productsにGETメソッドでアクセスするとJSONが返却される responseFromApiProductsByGETIsJSON()
  - api/productsにGETメソッドでアクセスして返却されるJSONは要件通りである JSONFromApiProductsByGETSatisfyRequirements()
  - api/productsにGETメソッドでアクセスして返却される商品情報は10件である ProductsCountFromApiProductsByGETIs10()
- api/shopsにGETメソッドでアクセスするとJSONで店舗情報が5件返却される.より粒度を細かくすると以下の通り.
  - api/shopsにGETメソッドでアクセスするとJSONが返却される responseFromApiShopsByGETIsJSON()
  - api/shopsにGETメソッドでアクセスして返却されるJSONは要件通りである JSONFromApiShopsByGETSatisfyRequirements()
  - api/shopsにGETメソッドでアクセスして返却される店舗情報は5件である ProductsCountFromApiShopsByGETIs5()

finished all at 7/25 11:00

5. POSTメソッドを用いてデータ追加を行う機能を検証する.要件は以下の通り.
   - api/productsに[商品タイトル,商品説明,商品価格]をPOSTすると、productsテーブルにそのデータが追加される canAddDataIntoProductsTableToAccessApiProductsByPOST()
   - api/shopsに[店舗名称]をPOSTすると、shopsテーブルにそのデータが追加される canAddDataIntoShopsTableToAccessApiShopsByPOST()

   finished all at 7/25 14:00
6. POSTメソッドにValidationを実装する.また,validation用のテストを追加する.
   - api/productsにPOSTメソッドでデータを送信した際に,'title'が含まれない場合 422UnprocessableEntityが返却される. responseFromApiProductsWithoutTitleIs422ByPOST()
   - api/productsにPOSTメソッドでデータを送信した際に,'description'が含まれない場合 422UnprocessableEntityが返却される. responseFromApiProductsWithoutDescriptionIs422ByPOST()
   - api/productsにPOSTメソッドでデータを送信した際に,'price'が含まれない場合 422UnprocessableEntityが返却される. responseFromApiProductsWithoutPriceIs422ByPOST()
   - api/productsにPOSTメソッドでデータを送信した際に,それが空JSONデータの時 422UnprocessableEntityが返却される. responseFromApiShopsWithoutAllIs422ByPOST()
   - api/shopsにPOSTメソッドでデータを送信した際に,'name'が含まれない場合 422UnprocessableEntityが返却される.

responseFromApiShopsWithoutNameIs422ByPOST()

finished all at 7/25 14:40

7. api/productsとapi/shopsに対してGET,POSTでアクセスした際の処理を,ApiProductController, ApiShopControllerに移動させる.
    - 各Controllerに処理を移譲したのち,上記のテストが破綻しなければよい.それぞれ関数名は以下のようにする.
        - api/productsに対するGETメソッド
          ApiProductController@getProducts
        - api/productsに対するPOSTメソッド
          ApiProductController@addProduct
        - api/shopsに対するGETメソッド
          ApiShopController@getShops
        - api/shopsに対するPOSTメソッド
          ApiShopController@addShop

    finished all at 7/25 15:00
8. 上記7において実装した各メソッドをさらにServiceクラスとControllerクラスに分離する.仕様は以下の通り.
    - api/productsに対するGETメソッドに対応するServiceクラスメソッド
      getProducts()
    - api/productsに対するPOSTメソッドに対応するServiceクラスメソッド
      addProduct()
    - api/shopsに対するGETメソッドに対応するServiceクラスメソッド
      getShops()
    - api/shopsに対するPOSTメソッドに対応するServiceクラスメソッド
      addShop()

    finished all at 7/25 19:00