PHP Ernest Code Challenge

Approach

My approach to this problem is to separate concerns using various design patterns i.e TDD and DDD.

Core of the system

The solution was built following SOLID principles. The solution will be split into 2 domains i.e the calculator domain and the unit convertor domain. They will both be dependents in the processDistanceAddition domain. The API will need to be versioned, so the endpoints will be prefixed with the version number as well as the controllers.

Request life cycle

Request inputs

Request	Data type
unit1	string
unit2	string
value1	float
value2	float
return_unit	string

Validation

FormRequest class

To make the controller slim, validation is done in the FormRequest class and before the request reaches the controller.

Once the request hits the controller it should go to a class that processes the request, by instantiating the unitConvertor and calculator class.

On success from the processor class we get the result and format the result as json and send the response to the client.

On failure, exceptions thrown will be caught and formatted as json and send the response to the client.

Unit conversion

This class will be responsible for converting value from one unit to a base unit given a value. This class must allow extension i.e converting more units of measurement.

Calculator

This class will be responsible for adding the given values.This class must allow extension i.e. more operations eg subtraction.

ProcessLength

This class will depend on the two above classes and get the final result.