Project Overview

I created a project with Laravel. First, I set up the database and the server.

Problem Analysis

According to the task, which included 2 example requests (it doesn't matter if they are from the frontend or API), we treat the system as an intermediary. So, all inputs, which are often messy (for example, using both _ and camelCase), need to be accepted by our system.

To handle this, I needed a command file: **InsuranceRequestCommand**. Next, I had to create a command that would take the input files and pass them to this command.

After reading the request contents, I filtered them for security reasons to find the expected values. Then, using an enum and a few other operators, I converted the inputs (also called user inputs) into a **DTO (Data Transfer Object)**, which would be more readable and understandable across the application.

DTOs

The **InsuranceRequestDTO** contains only the fields specified in the task.

After filtering the inputs and converting them into a shared language for the application, the data is sent to a service. This service determines which provider to communicate with. By creating an interface, we have the ability to implement services with different behaviors (polymorphism). It's important to keep in mind which service the interface refers to.

I implemented this with the **Strategy Design Pattern** using the file **InsuranceServiceStrategy**.

The system then communicates with the provider based on the settings in the .env file.

Finally, the DTO is passed to the service, and a mapper converts it into the format needed by the provider.

This structure allows each provider to have a custom implementation.

Result Handling

According to the problem statement, I only show the transformed request as text in the output. However, logically, this is not correct. The provider's result should be mapped back and returned to the service as a DTO. The insurance service should then return it to the appropriate controller, which would give the response as a resource in a defined format.

Testing

In this section, I focused on unit tests for the main components:

- The mapping process and converting data into the appropriate output format for each provider.
- Testing the **InsuranceServiceStrategy** was done because it depends on infrastructure.