## Product (Computer)

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Computer extends Model
{
    use HasFactory;

    protected $fillable = ['cpu', 'ram', 'storage'];

    // Usa casting para convertir el campo `parts` en un array automáticamente
    protected $casts = [
        'parts' => 'array',
    ];
}
```

## Builder

```php
<?php

namespace App\Patron;

/**
 * La interfaz Builder especifica métodos para crear las diferentes partes del computador.
 */
interface Builder
{
    public function buildCPU(): void;

    public function buildRAM(): void;

    public function buildStorage(): void;
}
```

# ConcreteBuilder (Computer Builder):

```php
<?php
namespace App\Patron;


use App\Models\Computer;
use App\Patron\Builder;
use App\Patron\Director;

class ComputerBuilder implements Builder
{
    private $computer;

    public function __construct()
    {
        $this->reset();
    }

    public function reset(): void
    {
        $this->computer = new Computer();
        $this->computer->parts = [];  // Inicializa el array de parts
    }

    public function buildCPU(): void
    {
        $parts = $this->computer->parts;
        $parts[] = "Intel i7 CPU";
        $this->computer->parts = $parts;
    }

    public function buildRAM(): void
    {
        $parts = $this->computer->parts;
        $parts[] = "16GB RAM";
        $this->computer->parts = $parts;
    }

    public function buildStorage(): void
    {
        $parts = $this->computer->parts;
        $parts[] = "512GB SSD";
        $this->computer->parts = $parts;
    }

    public function getComputer(): Computer
    {
        $result = $this->computer;
        $this->reset();

        return $result;
    }
}
```
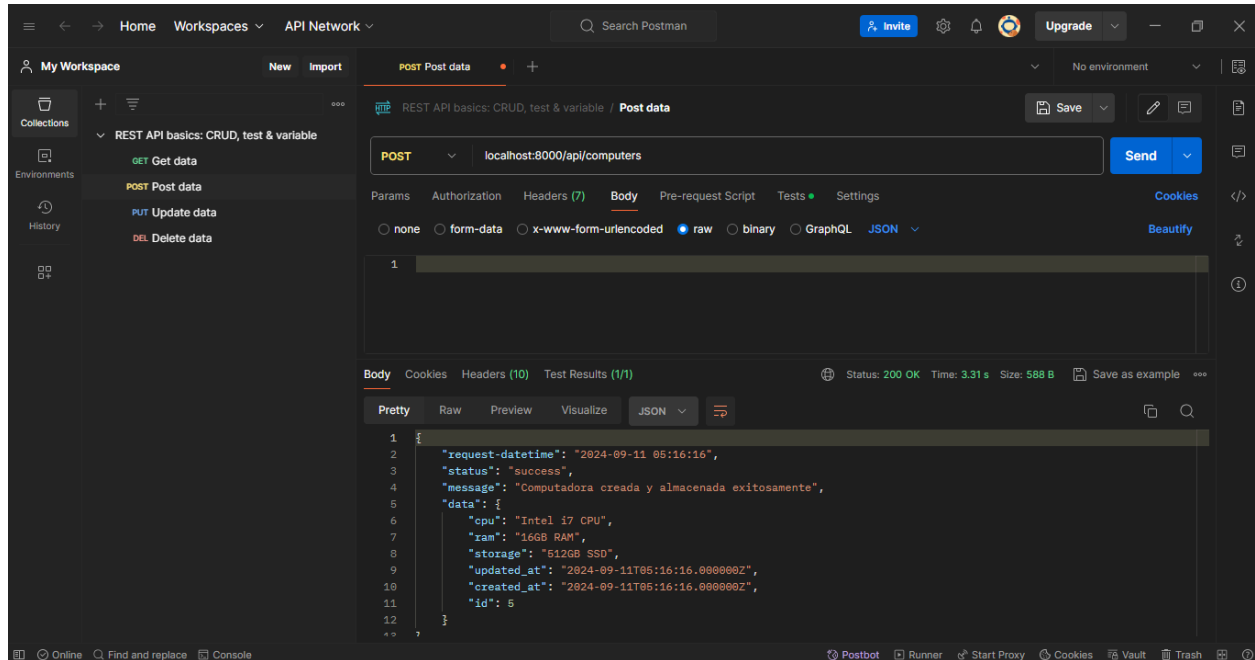
# Director:

```php
<?php

namespace App\Patron;

use App\Patron\Builder;

class Director
{
    private $builder;

    public function setBuilder(Builder $builder): void
    {
        $this->builder = $builder;
    }

    public function buildBasicComputer(): void
    {
        $this->builder->buildCPU();
    }

    public function buildHighEndComputer(): void
    {
        $this->builder->buildCPU();
        $this->builder->buildRAM();
        $this->builder->buildStorage();
    }
}
```

# ClientCode (Computer Controller):

```php
<?php

namespace App\Http\Controllers;


use Illuminate\Http\Request;
use App\Models\Computer;
use App\Patron\ComputerBuilder;
use App\Patron\Director;

class ComputerController extends Controller
{
    public function store(Request $request)
    {
        $director = new Director();
        $builder = new ComputerBuilder();
        $director->setBuilder($builder);

        // Construcción de un computador de alta gama (High-End)
        $director->buildHighEndComputer();

        // Para computador básico, usa esta línea en su lugar:
        //$director->buildBasicComputer();

        // Para un computador personalizado, puedes usar esto directamente:
        //$builder->buildCPU();
        //$builder->buildStorage();
```

```php
        $computer = $builder->getComputer();

        // Guardar en la base de datos
        $newComputer = Computer::create([
            'cpu' => $computer->parts[0] ?? null,
            'ram' => $computer->parts[1] ?? null,
            'storage' => $computer->parts[2] ?? null,
        ]);

        return response()->json([
            'request-datetime' => Date('Y-m-d H:i:s'),
            'status' => 'success',
            'message' => 'Computadora creada y almacenada exitosamente',
            'data' => $newComputer
        ]);
    }
}
```

# Resultado de la ejecución (Postman):