Solucion de Inscripciones Form Validation, Service Layer, Repository Pattern, Métodos limpios en controlador.

Implementa el patron Repository a los modelos existentes, form validations y una capa de servicio que permita crear todo lo necesario para la inscripción según como te muestro abajo.

En el controlador InscripcionesController implementa como te muestro en el ejemplo.

FORM VALIDATION:

```
// App/Http/Requests/InscripcionRequest.php

class InscripcionRequest extends FormRequest

{

   public function rules()

   {

      return [

         'alumno.nombre' => 'required|string|max:255',

         'alumno.dni' => 'required|string|unique:alumnos,dni',

         'tutores.*.nombre' => 'required|string',

         'domicilio.calle' => 'required|string',

         // ... todas tus reglas

      ];

   }

}
```

SERVICE LAYER

```
// App/Services/InscripcionService.php

class InscripcionService

{

   public function __construct(

      private AlumnoRepository $alumnoRepository,

      private TutorRepository $tutorRepository,

      private DomicilioRepository $domicilioRepository

   ) {}


   public function crearInscripcion(array $datos): Inscripcion
```

```
{
    return DB::transaction(function () use ($datos) {
        // Buscar o crear alumno
        $alumno = $this->procesarAlumno($datos['alumno']);

        // Procesar tutores
        $tutores = $this->procesarTutores($datos['tutores'], $alumno);

        // Procesar domicilio
        $domicilio = $this->procesarDomicilio($datos['domicilio'], $alumno);

        // Crear inscripción
        return $alumno->inscripciones()->create([
            'ciclo_lectivo' => $datos['ciclo_lectivo'],
            'curso_id' => $datos['curso_id'],
            // ... otros datos
        ]);
    });
}

private function procesarAlumno(array $datosAlumno): Alumno
{
    if (isset($datosAlumno['id'])) {
        return $this->alumnoRepository->actualizar(
            $datosAlumno['id'],
            $datosAlumno
        );
    }

    return $this->alumnoRepository->crear($datosAlumno);
}
```

```
}
```

REPOSITORY PATTERN

```
// App/Repositories/AlumnoRepository.php
class AlumnoRepository
{
    public function crear(array $datos): Alumno
    {
        return Alumno::create($datos);
    }

    public function actualizar(int $id, array $datos): Alumno
    {
        $alumno = Alumno::findOrFail($id);
        $alumno->update($datos);
        return $alumno->fresh();
    }

    public function buscarPorDni(string $dni): ?Alumno
    {
        return Alumno::where('dni', $dni)->first();
    }
}
```

INSCRIPCIONES CONTROLLER

```
class InscripcionController extends Controller
{
    public function __construct(
        private InscripcionService $inscripcionService
    ) {}

    public function store(InscripcionRequest $request)
```

```
    {
        try {
            $inscripcion = $this->inscripcionService->crearInscripcion(
                $request->validated()
            );

            return redirect()
                ->route('inscripciones.show', $inscripcion)
                ->with('success', 'Inscripción creada exitosamente');

        } catch (\Exception $e) {
            return back()
                ->withInput()
                ->withErrors(['error' => 'Error al crear la inscripción']);
        }
    }
}
```

Ten en cuenta que son ejemplos únicamente, la estructura real de los datos será tomada de los modelos de este proyecto.