1. Diseño de la Base de Datos

Se necesitan varias tablas para representar las relaciones entre administradores, grupos de pre-IPO, gestores y clientes. Aquí hay un esquema básico:

- `users`: Tabla para todos los usuarios del sistema (administradores, gestores, clientes).
  - `id`
  - `name`
  - `email`
  - `password`
  - `role` (admin, manager, client)

- `preipo_groups`: Tabla para los grupos de pre-IPO.
  - `id`
  - `name`
  - `description`
  - `created_by` (foreign key a `users.id`)

- `managers`: Tabla para los gestores de banca.
  - `id`
  - `user_id` (foreign key a `users.id`)
  - `bank_name`

- `clients`: Tabla para los clientes.
  - `id`
  - `user_id` (foreign key a `users.id`)

- `manager_id` (foreign key a `managers.id`)

- `group_manager`: Tabla pivot para la relación muchos a muchos entre grupos de pre-IPO y gestores.
  - `id`
  - `group_id` (foreign key a `preipo_groups.id`)
  - `manager_id` (foreign key a `managers.id`)

- `investments`: Tabla para las inversiones de los clientes.
  - `id`
  - `client_id` (foreign key a `clients.id`)
  - `group_id` (foreign key a `preipo_groups.id`)
  - `amount`
  - `status

2. Modelos de Eloquent

Se deben crear modelos para cada tabla mencionada:

```php
// app/Models/User.php
namespace App\Models;

use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;

class User extends Authenticatable
{
    use Notifiable;
```

```
    protected $fillable = [
        'name', 'email', 'password', 'role',
    ];

    protected $hidden = [
        'password', 'remember_token',
    ];

    public function manager()
    {
        return $this->hasOne(Manager::class);
    }

    public function client()
    {
        return $this->hasOne(Client::class);
    }
}

// app/Models/PreipoGroup.php
namespace App\Models;

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

class PreipoGroup extends Model
{
```

```
    use HasFactory;

    protected $fillable = ['name', 'description', 'created_by'];

    public function managers()
    {
        return $this->belongsToMany(Manager::class, 'group_manager');
    }
}

// app/Models/Manager.php
namespace App\Models;

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

class Manager extends Model
{
    use HasFactory;

    protected $fillable = ['user_id', 'bank_name'];

    public function user()
    {
        return $this->belongsTo(User::class);
    }

    public function clients()
```

```
    {
        return $this->hasMany(Client::class);
    }

    public function groups()
    {
        return $this->belongsToMany(PreipoGroup::class, 'group_manager');
    }
}

// app/Models/Client.php
namespace App\Models;

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

class Client extends Model
{
    use HasFactory;

    protected $fillable = ['user_id', 'manager_id'];

    public function user()
    {
        return $this->belongsTo(User::class);
    }

    public function manager()
```

```
    {
        return $this->belongsTo(Manager::class);
    }

    public function investments()
    {
        return $this->hasMany(Investment::class);
    }
}

// app/Models/Investment.php
namespace App\Models;

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

class Investment extends Model
{
    use HasFactory;

    protected $fillable = ['client_id', 'group_id', 'amount', 'status'];

    public function client()
    {
        return $this->belongsTo(Client::class);
    }

    public function group()
```

```
    {
        return $this->belongsTo(PreipoGroup::class);
    }
}
```

3. Controladores

Los controladores manejarán las operaciones CRUD para los diferentes modelos. Aquí hay un ejemplo para el controlador de grupos de pre-IPO:

```php
// app/Http/Controllers/PreipoGroupController.php
namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Models\PreipoGroup;
use App\Models\Manager;

class PreipoGroupController extends Controller
{
    public function index()
    {
        $groups = PreipoGroup::all();
        return view('preipo_groups.index', compact('groups'));
    }

    public function create()
```

```
    {
        return view('preipo_groups.create');
    }

    public function store(Request $request)
    {
        $validated = $request->validate([
            'name' => 'required|string|max:255',
            'description' => 'nullable|string',
        ]);

        $group = PreipoGroup::create([
            'name' => $validated['name'],
            'description' => $validated['description'],
            'created_by' => auth()->id(),
        ]);

        return redirect()->route('preipo_groups.index')->with('success', 'Group created
successfully.');
    }

    public function show(PreipoGroup $preipoGroup)
    {
        return view('preipo_groups.show', compact('preipoGroup'));
    }

    public function edit(PreipoGroup $preipoGroup)
    {
```

```
        return view('preipo_groups.edit', compact('preipoGroup'));
    }

    public function update(Request $request, PreipoGroup $preipoGroup)
    {
        $validated = $request->validate([
            'name' => 'required|string|max:255',
            'description' => 'nullable|string',
        ]);

        $preipoGroup->update($validated);

        return redirect()->route('preipo_groups.index')->with('success', 'Group updated successfully.');
    }

    public function destroy(PreipoGroup $preipoGroup)
    {
        $preipoGroup->delete();
        return redirect()->route('preipo_groups.index')->with('success', 'Group deleted successfully.');
    }

    public function addManager(Request $request, PreipoGroup $preipoGroup)
    {
        $validated = $request->validate([
            'manager_id' => 'required|exists:managers,id',
        ]);
```

```
        $preipoGroup->managers()->attach($validated['manager_id']);

        return redirect()->route('preipo_groups.show', $preipoGroup)->with('success', 'Manager added successfully.');
    }
}
```

4. Rutas

Definir las rutas necesarias en `routes/web.php`:

```php
use App\Http\Controllers\PreipoGroupController;

Route::middleware(['auth'])->group(function () {
    Route::resource('preipo_groups', PreipoGroupController::class);
    Route::post('preipo_groups/{preipoGroup}/add_manager', [PreipoGroupController::class, 'addManager'])->name('preipo_groups.add_manager');
});
```

5. Vistas

Crear las vistas necesarias en `resources/views/preipo_groups/` para las operaciones CRUD y la gestión de gestores.

6. Middleware y Autorización

Agregar middleware para garantizar que solo los administradores puedan crear y gestionar grupos de pre-IPO:

```php
// app/Http/Middleware/CheckAdmin.php
namespace App\Http\Middleware;

use Closure;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;

class CheckAdmin
{
    public function handle(Request $request, Closure $next)
    {
        if (Auth::check() && Auth::user()->role === 'admin') {
            return $next($request);
        }

        return redirect('/')->with('error', 'Access denied.');
    }
}

// En Kernel.php, registrar el middleware
protected $routeMiddleware = [
    // ...
    'admin' => \App\Http\Middleware\CheckAdmin::class,
```

```
];
```