FASE 2 - MEJORAS DE ALTA PRIORIDAD

Resumen de Cambios Implementados

Fecha: 4 de Octubre, 2025
Rama: `feat/fase2-prioridad`
Base: `feat/fase1-notificaciones-localidad-ruta`

📋 ÍNDICE DE CAMBIOS

1. VALIDACIONES DE NEGOCIO ✅

Request Classes Creadas:

- **`StoreRecoleccionOrganicosRequest.php`**
- Validación: No permite selección de fecha (se asigna automáticamente)
- Validación: Solo una recolección orgánica por semana por usuario
- Campos requeridos: `user_id`, `company_id`, `localidad_id`

- **`StoreRecoleccionInorganicosRequest.php`**

- Validación: Máximo 2 recolecciones por semana para tipo “programada”
- Validación: No permite duplicados en la misma fecha
- Validación: Fecha debe ser hoy o posterior
- Campos requeridos: `user_id`, `company_id`, `localidad_id`, `fecha_programada`, `tipo_recoleccion`
- Tipos de recolección: `programada` o `demanda`

- **`StoreRecoleccionPeligrososRequest.php`**

- Validación: Solo una solicitud por mes por usuario
- Validación: No permite nueva solicitud si hay una activa (solicitado, aprobado, programado)
- Campos requeridos: `user_id`, `company_id`, `localidad_id`, `descripcion_residuos`
- Campos opcionales: `cantidad_estimada`, `notas`

Modelo Collection - Métodos de Validación:

```
Collection::tieneSolicitudPeligrososMesActual($userId)
Collection::existeDuplicado($userId, $tipoResiduo, $fechaProgramada)
```

---

2. MÓDULO RECOLECCIONES ORGÁNICOS ✅

Características Implementadas:

- **Programación Automática Semanal**: La fecha se asigna automáticamente según la ruta de la localidad
- **Asignación Automática de Ruta**: Se obtiene la ruta activa de la localidad del usuario

- **Cálculo de Próxima Fecha**: Método `getProximaFechaRecoleccion()` en modelo Ruta

Archivos Modificados/Creados:

Modelo Ruta (`app/Models/Ruta.php`):
- Método `getProximaFechaRecoleccion()` : Calcula la próxima fecha según día de semana
- Método `getFechasRecoleccionProximasSemanas($semanas)` : Obtiene fechas para N semanas
- Scopes: `activas()` , `porDiaSemana($dia)` , `porLocalidad($localidadId)`
- Constante `DIAS_SEMANA` : Mapeo de días (1=Lunes, 7=Domingo)

CollectionController (`app/Http/Controllers/Web/CollectionController.php`):
- Método `storeOrganicos(Request $request)` : Crea recolección orgánica automática
- Valida que no exista recolección esta semana
- Obtiene ruta de la localidad
- Asigna fecha automáticamente
- Envía notificación por email

Comando Artisan (`app/Console/Commands/GenerarRecoleccionesOrganicosSemanales.php`):

```
php artisan recolecciones:generar-organicos-semanales --semanas=1 --dry-run
```

- Genera recolecciones para todos los usuarios activos
- Asigna automáticamente según localidad y ruta
- Opciones:
- `--semanas=N` : Número de semanas a generar (default: 1)
- `--dry-run` : Simular sin crear registros
- Programado en scheduler: Lunes 6:00 AM

Scheduler (`routes/console.php`):

```
Schedule::command('recolecciones:generar-organicos-semanales --semanas=1')-
>weeklyOn(1, '06:00');
```

---

3. MÓDULO RECOLECCIONES INORGÁNICOS ✅

Características Implementadas:

- **Dos Modalidades**: Programada (1-2 veces/semana) o Por Demanda
- **Validación de Frecuencia**: Máximo 2 recolecciones programadas por semana
- **Selección de Fecha**: Usuario puede elegir fecha (a diferencia de orgánicos)
- **Validación de Duplicados**: No permite recolecciones en la misma fecha

Archivos Modificados:

CollectionController:
- Método `storeInorganicos(Request $request)` : Crea recolección inorgánica
- Valida frecuencia semanal
- Valida duplicados
- Permite selección de fecha
- Envía notificación por email

---

4. MÓDULO RECOLECCIONES PELIGROSOS ✅

Características Implementadas:

- **Sistema de Solicitudes**: Flujo completo de solicitud → aprobación → programación
- **Estados de Solicitud**:
- `solicitado` : Solicitud inicial del usuario
- `aprobado` : Aprobada por administrador
- `rechazado` : Rechazada por administrador
- `programado` : Recolección programada después de aprobación
- `completado` : Recolección realizada

Migración (`database/migrations/2025_10_04_221936_add_estado_solicitud_to_collections.php`):

```
- estado_solicitud (string, nullable)
- fecha_solicitud (timestamp, nullable)
- fecha_aprobacion (timestamp, nullable)
- aprobado_por (foreign key users, nullable)
- motivo_rechazo (text, nullable)
```

Modelo Collection - Nuevas Constantes y Métodos:

```
// Constantes
const ESTADO_SOLICITADO = 'solicitado';
const ESTADO_APROBADO = 'aprobado';
const ESTADO_RECHAZADO = 'rechazado';
const ESTADO_PROGRAMADO = 'programado';
const ESTADO_COMPLETADO = 'completado';

// Scopes
scopePeligrosos($query)
scopeSolicitudesPendientes($query)
scopeEstadoSolicitud($query, $estado)

// Relaciones
aprobador() // Usuario que aprobó/rechazó
```

SolicitudPeligrososController (`app/Http/Controllers/SolicitudPeligrososController.php`):

Métodos para Usuarios:
- `index()` : Listar solicitudes del usuario
- `create()` : Formulario de nueva solicitud (valida límite mensual)
- `store()` : Crear solicitud
- `show($id)` : Ver detalle de solicitud
- `edit($id)` : Editar solicitud (solo si está en estado “solicitado”)
- `update($id)` : Actualizar solicitud
- `destroy($id)` : Cancelar solicitud

Métodos para Administradores:
- `pendientes()` : Listar todas las solicitudes pendientes
- `aprobar($id)` : Aprobar solicitud
- `rechazar($id)` : Rechazar solicitud (requiere motivo)
- `programar($id)` : Programar recolección después de aprobar

Rutas Agregadas (`routes/web.php`):

```
// Usuarios
Route::resource('solicitudes-peligrosos', SolicitudPeligrososController::class);

// Administradores
Route::get('/admin/solicitudes-peligrosos/pendientes', ...);
Route::post('/admin/solicitudes-peligrosos/{id}/aprobar', ...);
Route::post('/admin/solicitudes-peligrosos/{id}/rechazar', ...);
Route::post('/admin/solicitudes-peligrosos/{id}/programar', ...);
```

---

5. LÓGICA DE PUNTOS MEJORADA ✅

Migración (`database/migrations/2025_10_04_221935_add_requisitos_separacion_to_peso_registros.php`):

```
- requisitos_separacion (boolean, default: false)
- observaciones_separacion (text, nullable)
```

PuntosService (`app/Services/PuntosService.php`):

Métodos Principales:

- `calcularPuntos(CollectionDetail $pesoRegistro)` : Calcula puntos según peso
- Solo si `requisitos_separacion = true`
- Solo para tipo `INORGANICO`
- Fórmula: 1 kg = 1 punto (preparado para Strategy Pattern en Fase 3)

- `asignarPuntos(User $user, float $puntos)` : Asigna puntos al usuario
- Usa el modelo Point existente
- Registra en logs

- `procesarPuntos(CollectionDetail $pesoRegistro)` : Proceso completo
- Calcula y asigna puntos automáticamente
- Retorna resultado con mensaje

- `puedeGenerarPuntos(CollectionDetail $pesoRegistro)` : Validación
- Verifica requisitos de separación
- Verifica tipo de residuo
- Verifica peso > 0

Modelo CollectionDetail - Eventos Automáticos:

```
protected static function boot()
{
    parent::boot();

    // Evento: created
    // Procesa puntos automáticamente al crear registro

    // Evento: updated
    // Procesa puntos si se actualiza requisitos_separacion a true
}
```

Campos Agregados al Fillable:

- `requisitos_separacion`
- `observaciones_separacion`

---

6. MODELO COLLECTION - MEJORAS ✅

Constantes Agregadas:

```
// Tipos de residuo
const TIPO_ORGANICO = 'FO';
const TIPO_INORGANICO = 'INORGANICO';
const TIPO_PELIGROSO = 'PELIGROSO';

// Estados de solicitud
const ESTADO_SOLICITADO = 'solicitado';
const ESTADO_APROBADO = 'aprobado';
const ESTADO_RECHAZADO = 'rechazado';
const ESTADO_PROGRAMADO = 'programado';
const ESTADO_COMPLETADO = 'completado';
```

Nuevos Scopes:

- `organicos()`: Filtrar recolecciones orgánicas
- `inorganicos()`: Filtrar recolecciones inorgánicas
- `peligrosos()`: Filtrar recolecciones peligrosas
- `estadoSolicitud($estado)`: Filtrar por estado de solicitud
- `solicitudesPendientes()`: Solicitudes en estado “solicitado”

Nuevas Relaciones:

- `aprobador()`: Usuario que aprobó/rechazó la solicitud

Métodos Estáticos de Validación:

- `tieneSolicitudPeligrososMesActual($userId)`: Verifica límite mensual
- `existeDuplicado($userId, $tipoResiduo, $fechaProgramada)`: Verifica duplicados

📊 RESUMEN DE ARCHIVOS

Archivos Creados (13):

1. `app/Http/Requests/StoreRecoleccionOrganicosRequest.php`
2. `app/Http/Requests/StoreRecoleccionInorganicosRequest.php`
3. `app/Http/Requests/StoreRecoleccionPeligrososRequest.php`
4. `app/Services/PuntosService.php`
5. `app/Console/Commands/GenerarRecoleccionesOrganicosSemanales.php`
6. `app/Http/Controllers/SolicitudPeligrososController.php`
7. `database/migrations/2025_10_04_221935_add_requisitos_separacion_to_peso_registros.php`
8. `database/migrations/2025_10_04_221936_add_estado_solicitud_to_collections.php`
9. `FASE2_RESUMEN_CAMBIOS.md` (este archivo)

Archivos Modificados (6):

1. `app/Models/Collection.php` - Scopes, constantes, validaciones
2. `app/Models/Ruta.php` - Métodos de fecha, scopes
3. `app/Models/CollectionDetail.php` - Eventos automáticos, puntos
4. `app/Http/Controllers/Web/CollectionController.php` - Métodos orgánicos e inorgánicos
5. `routes/web.php` - Rutas nuevas
6. `routes/console.php` - Scheduler para orgánicos

🔄 FLUJOS DE TRABAJO

Flujo: Recolección de Orgánicos

1. Usuario solicita recolección orgánica (sin elegir fecha)
2. Sistema valida que no exista recolección esta semana
3. Sistema obtiene ruta de la localidad del usuario
4. Sistema calcula próxima fecha según día de ruta
5. Sistema crea recolección automáticamente
6. Sistema envía email de confirmación

Flujo: Recolección de Inorgánicos

1. Usuario elige fecha y tipo (programada/demanda)
2. Sistema valida frecuencia (máx 2/semana si es programada)
3. Sistema valida que no exista duplicado en esa fecha
4. Sistema crea recolección
5. Sistema envía email de confirmación

Flujo: Solicitud de Peligrosos

1. Usuario crea solicitud (descripción de residuos)
2. Sistema valida límite mensual (1 por mes)
3. Sistema crea solicitud en estado “solicitado”
4. Administrador revisa solicitudes pendientes
5. Administrador aprueba o rechaza (con motivo si rechaza)

6. Si aprobada: Administrador programa fecha y ruta
7. Sistema actualiza estado a “programado”
8. Recolección se realiza normalmente

Flujo: Asignación de Puntos

1. Recolector registra peso en CollectionDetail
2. Recolector marca `requisitos_separacion = true` si cumple
3. Evento `created` se dispara automáticamente
4. PuntosService calcula puntos (solo si INORGANICO y requisitos=true)
5. PuntosService asigna puntos al usuario
6. Sistema registra en logs

🧪 COMANDOS DE PRUEBA

Generar Recolecciones Orgánicos (Dry Run):

```
php artisan recolecciones:generar-organicos-semanales --dry-run
```

Generar Recolecciones Orgánicos (Real):

```
php artisan recolecciones:generar-organicos-semanales --semanas=1
```

Generar para 4 semanas:

```
php artisan recolecciones:generar-organicos-semanales --semanas=4
```

Verificar Migraciones:

```
php artisan migrate --pretend
```

Ejecutar Migraciones:

```
php artisan migrate
```

📝 NOTAS IMPORTANTES

Requisitos de Separación:

- Solo los residuos **INORGÁNICOS** generan puntos
- Solo si `requisitos_separacion = true`
- Fórmula actual: **1 kg = 1 punto**
- Preparado para Strategy Pattern en Fase 3

Límites y Validaciones:

- **Orgánicos**: 1 recolección por semana (automática)
- **Inorgánicos**: 2 recolecciones por semana (programadas) + ilimitadas por demanda
- **Peligrosos**: 1 solicitud por mes

Scheduler:

- Comando de orgánicos programado para **Lunes 6:00 AM**
- Genera recolecciones para la semana siguiente
- Requiere configurar cron job en servidor:

```
* * * * * cd /path-to-project && php artisan schedule:run >> /dev/null 2>&1
```

Seguridad:

- Todas las rutas protegidas con middleware `auth`
- Rutas de administración protegidas con `role:Administrador`
- Validaciones en Request classes y modelos
- Logs de todas las operaciones críticas

🚀 PRÓXIMOS PASOS (FASE 3)

1. **Strategy Pattern para Puntos**: Diferentes fórmulas según tipo de residuo
2. **Notificaciones Push**: Además de email
3. **Dashboard de Estadísticas**: Para usuarios y administradores
4. **Gamificación**: Niveles, badges, rankings
5. **API REST**: Para aplicación móvil
6. **Tests Automatizados**: Unit tests y feature tests

👥 ROLES Y PERMISOS

Usuario Regular:

- Solicitar recolecciones (orgánicos, inorgánicos, peligrosos)
- Ver sus propias solicitudes
- Cancelar solicitudes pendientes
- Ver sus puntos

Administrador:

- Todo lo anterior +
- Ver todas las solicitudes de peligrosos
- Aprobar/rechazar solicitudes
- Programar recolecciones de peligrosos
- Gestionar localidades y rutas
- Ejecutar comandos artisan

Empresa Recolectora:

- Registrar pesos de recolecciones
- Marcar requisitos de separación
- Ver recolecciones asignadas

📧 CONTACTO Y SOPORTE

Para dudas o problemas con la implementación de Fase 2, contactar al equipo de desarrollo.

Fecha de Implementación: 4 de Octubre, 2025
Versión: 2.0.0
Estado: ✅ Completado y Listo para Testing