Vou explicar detalhadamente o fluxo de registro de resíduos:

1. \*\*Controller (WasteController.java)\*\*:

```java

@PostMapping("/register")

public String registerWaste(@ModelAttribute Waste waste,

@AuthenticationPrincipal UserDetails userDetails,

RedirectAttributes redirectAttributes) {

try {

// Define a data atual

waste.setDate(LocalDate.now());

// Salva o resíduo

wasteService.save(waste, userDetails.getUsername());

redirectAttributes.addFlashAttribute("message", "Resíduo registrado com sucesso!");

redirectAttributes.addFlashAttribute("messageType", "alert-success");

return "redirect:/dashboard";

} catch (Exception e) {

redirectAttributes.addFlashAttribute("message", "Erro ao registrar resíduo: " + e.getMessage());

redirectAttributes.addFlashAttribute("messageType", "alert-danger");

return "redirect:/waste/register";

}

}

```

- `@PostMapping("/register")`: Define que este método processa requisições POST para `/waste/register`

- `@ModelAttribute Waste waste`: Recebe os dados do formulário e converte automaticamente para um objeto `Waste`

- `@AuthenticationPrincipal UserDetails userDetails`: Obtém os dados do usuário autenticado

- `RedirectAttributes redirectAttributes`: Permite enviar mensagens para a próxima página após o redirecionamento

2. \*\*Service (WasteService.java)\*\*:

```java

public void save(Waste waste, String userEmail) {

User user = userRepository.findByEmail(userEmail)

.orElseThrow(() -> new RuntimeException("Usuário não encontrado"));

waste.setUser(user);

wasteRepository.save(waste);

}

```

- Recebe o objeto `Waste` e o email do usuário

- Busca o usuário no banco de dados pelo email

- Se não encontrar o usuário, lança uma exceção

- Associa o usuário ao resíduo

- Salva o resíduo no banco de dados

3. \*\*Fluxo completo\*\*:

a) \*\*Quando o usuário acessa a página\*\*:

- Acessa `/waste/register` (GET)

- O controller retorna o template `waste/register.html`

- O template exibe o formulário com campos para:

- Tipo de resíduo (select)

- Peso (number)

- Descrição (textarea)

b) \*\*Quando o usuário submete o formulário\*\*:

1. O formulário envia um POST para `/waste/register`

2. O Spring converte os dados do formulário para um objeto `Waste`

3. O controller:

- Define a data atual no objeto `Waste`

- Chama o `WasteService.save()` passando o resíduo e o email do usuário

4. O service:

- Busca o usuário pelo email

- Associa o usuário ao resíduo

- Salva no banco de dados

5. O controller:

- Se sucesso: redireciona para `/dashboard` com mensagem de sucesso

- Se erro: redireciona de volta para `/waste/register` com mensagem de erro

4. \*\*Tratamento de erros\*\*:

- Se o usuário não for encontrado: exibe "Usuário não encontrado"

- Se houver qualquer outro erro: exibe a mensagem do erro

- As mensagens são exibidas usando classes do Bootstrap:

- Sucesso: `alert-success` (verde)

- Erro: `alert-danger` (vermelho)

5. \*\*Segurança\*\*:

- O usuário precisa estar autenticado para acessar a página

- O email do usuário é obtido automaticamente do contexto de segurança

- O resíduo é sempre associado ao usuário que o criou

Este código implementa um padrão comum em aplicações Spring:

- Controller: lida com requisições HTTP e redirecionamentos

- Service: contém a lógica de negócio

- Repository: interage com o banco de dados

- Model: representa os dados

- Template: apresenta a interface ao usuário