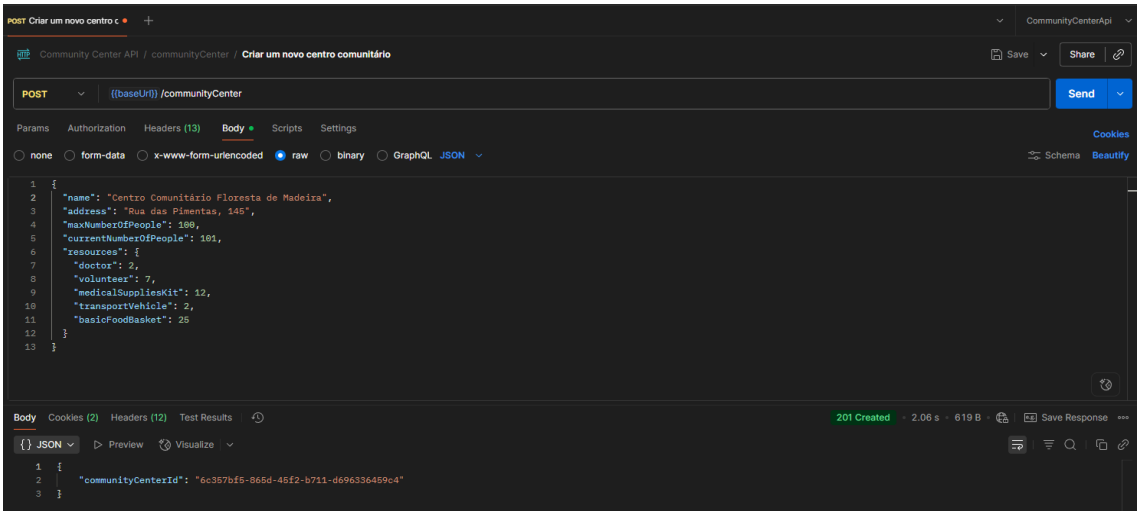
Relatório de Bugs e Sugestões de Melhoria - API de Centros Comunitários

1. Bugs Encontrados

A seguir, uma lista detalhada dos bugs identificados durante a fase de testes exploratórios e manuais da aplicação.

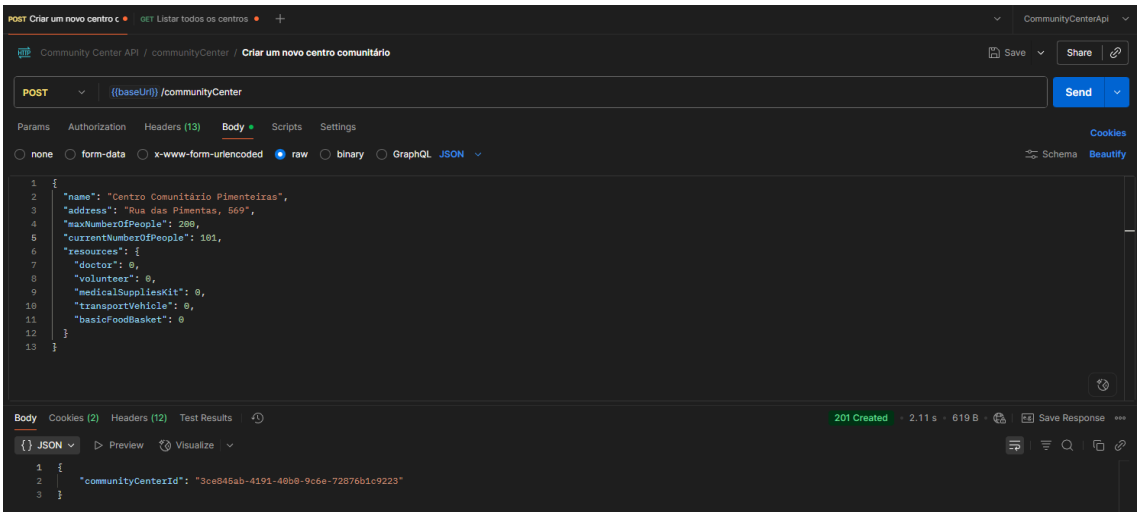
BUG-001: A API permite criar um Centro Comunitário mesmo que o limite máximo de pessoas seja atingido na criação

- **Título:** Validação de capacidade de ocupação falha ao criar um novo centro.
- **Passos para Reproduzir:**
    1. Realizar uma requisição POST para o endpoint /centros.
    2. No corpo da requisição, enviar um JSON com o campo currentNumberOfPeople maior que o campo maxNumberOfPeople.
    3. Exemplo: { "nome": "Centro Teste", "maxNumberOfPeople": 100, "currentNumberOfPeople": 101, "recursos": ["médico", "psicólogo"] }
- **Resultado Esperado:** A API deveria retornar um erro `400 Bad Request` com uma mensagem clara, informando que a quantidade de pessoas não pode exceder a capacidade de ocupação.
- **Resultado Atual:** A API retorna `201 Created` e cria o centro com dados inconsistentes.
- **Severidade:** Alta. A falha permite a criação de registros com dados inválidos que podem afetar a lógica de negócio e a integridade do sistema.
- **Evidência**



BUG-002: A API permite criar um Centro Comunitário com os recursos zerados

- **Título:** Criação de Centro Comunitário é permitida com lista de recursos vazia ou nula.
- **Passos para Reproduzir:**
    1. Realizar uma requisição POST para o endpoint `/centros`.
    2. No corpo da requisição, enviar um JSON com o campo `recursos` como um array vazio.
    3. Exemplo: `{ "nome": "Centro Vazio", "maxNumberOfPeople": 50, "currentNumberOfPeople": 10, "recursos": [] }`
- **Resultado Esperado:** A API deveria retornar um erro `400 Bad Request`, indicando que um centro deve ser criado com pelo menos um recurso, conforme a regra de negócio.
- **Resultado Atual:** A API retorna `201 Created` e cria o centro sem recursos.
- **Severidade:** Média. Embora não quebre o sistema, vai contra a regra de negócio de que todo centro deve oferecer algo à comunidade.
- **Evidência:**

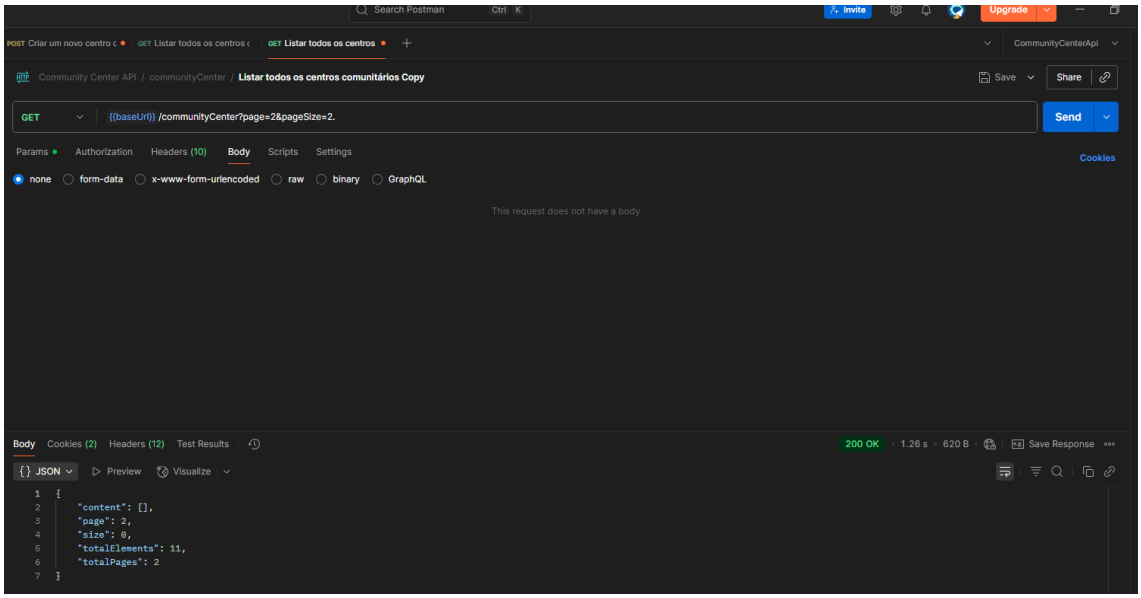


BUG-003: A listagem de centros não funciona com paginação

- **Título:** Endpoint de listagem de centros ignora parâmetros de paginação e retorna um array vazio.
- **Passos para Reproduzir:**
    1. Criar múltiplos centros comunitários (ex: 15 centros) para garantir que haja dados para paginar.

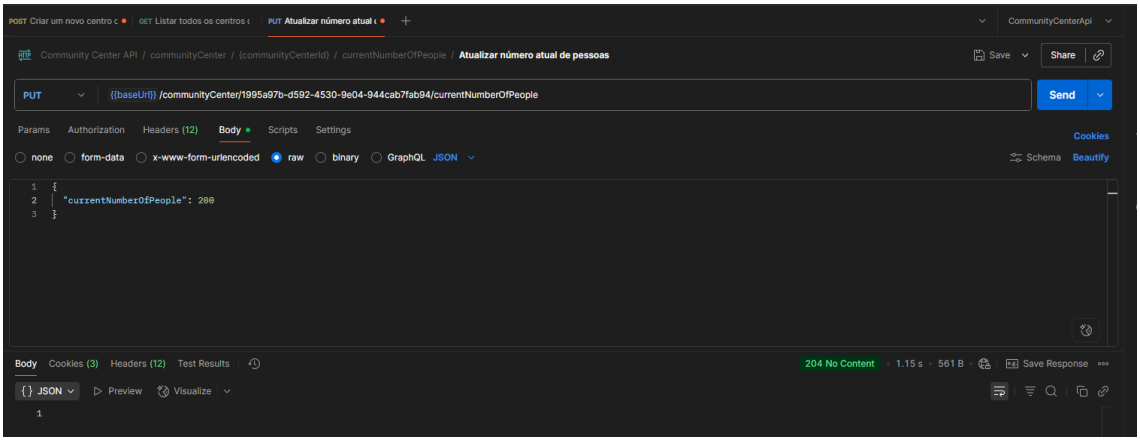
2. Realizar uma requisição `GET` para o endpoint `/centros` com parâmetros de paginação.
3. Exemplo: `GET /centros?page=1&limit=10`

- **Resultado Esperado:** A API deveria retornar um status `200 OK` com um objeto de paginação contendo os 10 primeiros centros cadastrados.
- **Resultado Atual:** A API retorna um status `200 OK` com um array vazio (`[]`), ignorando os dados existentes.
- **Severidade:** Crítica. Impede completamente a consulta dos dados da aplicação de forma escalável.
- **Evidência:**



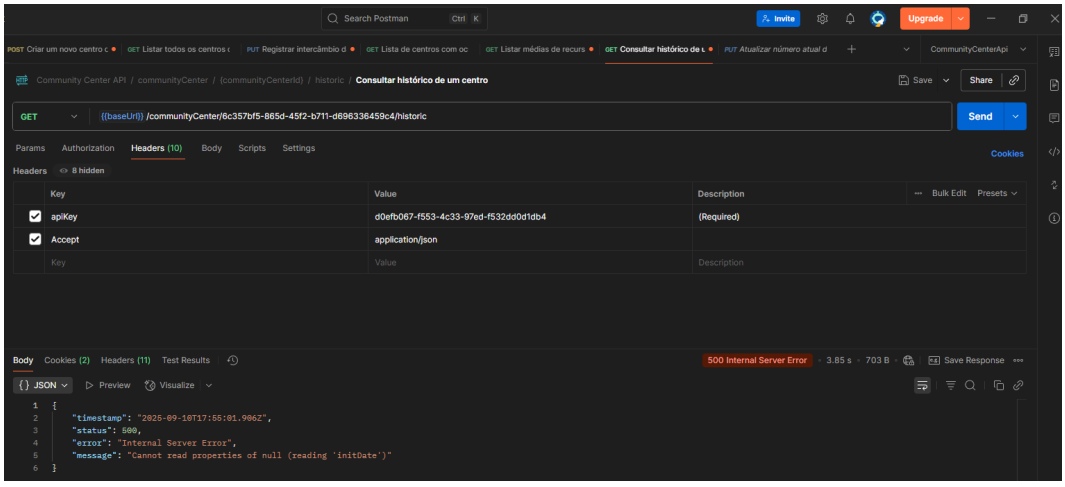
BUG-004: A API permite alterar a quantidade de pessoas além da capacidade de ocupação

- **Título:** Validação de capacidade de ocupação falha ao atualizar um centro existente.
- **Passos para Reproduzir:**
  1. Criar um centro com uma capacidade definida (ex: 100).
  2. Obter o ID do centro criado.
  3. Realizar uma requisição PUT para o endpoint /centros/{id}.
  4. No corpo da requisição, enviar um JSON atualizando o campo currentNumberOfPeople para um valor maior que a maxNumberOfPeople.
  5. Exemplo: { "currentNumberOfPeople": 150 }
- **Resultado Esperado:** A API deveria retornar um erro 400 Bad Request com uma mensagem clara, informando que a atualização viola a regra de capacidade.
- **Resultado Atual:** A API retorna 200 OK e atualiza o centro com o valor inconsistente.
- **Severidade:** Alta. Similar ao BUG-001, afeta a integridade dos dados e a lógica de negócio.
- **Evidência:**



BUG-005: A API retorna um erro 500 ao solicitar o histórico de um centro

- **Título:** Erro interno do servidor ao consultar o endpoint de histórico de um centro.
- **Passos para Reproduzir:**
    1. Criar um centro comunitário e obter seu ID.
    2. Realizar uma requisição GET para o endpoint `/centros/{id}/historic`.
- **Resultado Esperado:** A API deveria retornar um status `200 OK` com o histórico de ocupação e recursos do centro.
- **Resultado Atual:** A API retorna um status `500 Internal Server Error`, sem uma mensagem clara sobre a causa do erro.
- **Severidade:** Crítica. Uma funcionalidade principal da aplicação está indisponível e o erro não é tratado de forma adequada.
- **Evidência:**



2. Sugestões de Melhoria

1. **Melhorar a Validação de Entrada (Input Validation):**
    - Implementar validações mais robustas e centralizadas (Schema Validation) para todas as requisições POST e PUT. Isso evitaria a criação/atualização de dados

inconsistentes (Bugs 001, 002, 004) antes mesmo de a lógica de negócio ser processada. Ferramentas como Joi ou Zod podem ser usadas no backend para isso.

2. **Padronizar Mensagens de Erro:**
    - Evitar o retorno de erros genéricos como o `500 Internal Server Error` (Bug 005). Implementar um middleware de tratamento de erros para capturar exceções não tratadas e retornar uma resposta JSON padronizada, como `{ "statusCode": 500, "message": "Ocorreu um erro inesperado em nossos servidores. Tente novamente mais tarde." }`. Para erros de validação (`400`), a mensagem deve especificar qual campo está incorreto.
3. **Implementar Paginação Corretamente:**
    - A funcionalidade de paginação (Bug 003) é crucial para a performance e escalabilidade. A resposta deveria seguir um padrão de mercado, como:

```
{
  "data": [ ... ],
  "meta": {
    "totalItems": 100,
    "itemCount": 10,
    "itemsPerPage": 10,
    "totalPages": 10,
    "currentPage": 1
  }
}
```

4. **Documentação da API (Swagger/OpenAPI):**
    - Melhorar a documentação no arquivo `swagger.yaml` para incluir as regras de negócio, como a obrigatoriedade de recursos e os limites de valores. Detalhar também os possíveis códigos de erro para cada endpoint (200, 201, 400, 404, 500) e o schema de suas respostas.