Mapeamento de issues

CT 001 - Padronização de respostas e status codes

1. Identificação

Campo	Descrição
ID da Issue	CT-001
Data de registro	15/08/2025
Relator	Bruno Calazans
Sistema / Módulo	Usuários / API

2. Descrição

Campo	Descrição
Tipo de Issue	Melhoria
Descrição detalhada	Atualmente, a API retorna mensagens e status codes inconsistentes entre endpoints. Por exemplo, GET de usuário com ID inválido retorna `{ "id": "id deve ter exatamente 16 caracteres alfanuméricos" }` e status 400, enquanto outros endpoints retornam `{ "message": "..." }`. Isso dificulta a criação de testes padronizados e a integração com o front-end.

Evidência	Resposta JSON do GET /usuarios/:id com ID inválido: `{ "id": "id deve ter exatamente 16 caracteres alfanuméricos" }`

3. Impacto

Campo	Descrição
Gravidade	Alta
Impacto no sistema	Testes automatizados podem falhar; front-end precisa tratar múltiplos formatos de resposta; inconsistência na documentação e integração
Prioridade	Alta

4. Sugestões

Campo	Descrição
Solução proposta	Padronizar todas as respostas de erro da API usando o mesmo campo (`message`) e retornar status codes consistentes: 200 (sucesso), 400 (requisição inválida), 404 (não encontrado), 401 (não autorizado).
Mudanças sugeridas na API	Ajustar endpoints GET, POST, PUT e DELETE de usuários e produtos para retornarem mensagens

	uniformes e status codes consistentes.
Validações e testes	Validar com Postman: todos os endpoints de usuário e produto devem retornar JSON com campo `message` em caso de erro; status codes devem seguir o padrão definido.

5. Observações adicionais

- Essa melhoria facilita automação de testes e documentação da API.
- Pode ser aplicada também a endpoints de produtos e autenticação para consistência geral.

CT 002 - Validação de email

1. Identificação

Campo	Descrição
ID da Issue	ISS-002
Data de registro	15/08/2025
Relator	Bruno Calazans
Sistema / Módulo	Usuários / API

2. Descrição

Campo	Descrição
Tipo de Issue	Melhoria
Descrição detalhada	Atualmente, algumas regras de negócio, como bloqueio de e-mails de provedores Gmail/Hotmail e validação do tamanho da senha (5 a 10 caracteres), são apenas validadas nos testes do Postman, mas não

	implementadas no backend. Isso permite que requisições inválidas possam ser registradas diretamente no banco de dados.
Evidência	Testes de POST/PUT no Postman verificam e-mails bloqueados e senha inválida, mas a API aceita a requisição caso feita diretamente via CURL ou outro cliente.

3. Impacto

Campo	Descrição
Gravidade	Alta
Impacto no sistema	Inconsistência de dados, possível falha de segurança, regras de negócio não garantidas
Prioridade	Alta

4. Sugestões

Campo	Descrição
Solução proposta	Implementar validação no backend para: 1) impedir cadastro de usuários com e-mails dos provedores bloqueados; 2) validar tamanho da senha (5–10 caracteres); 3) retornar erro 400 com mensagem clara quando a regra não é atendida.
Mudanças sugeridas na API	Adicionar validações nos endpoints POST /usuarios e PUT /usuarios/:id; atualizar mensagens de erro padronizadas; impedir persistência de dados inválidos.
Validações e testes	Postman: criar testes automatizados para e-mails bloqueados, senhas inválidas e validação de mensagens de erro; verificar

	que o backend rejeita a requisição mesmo sem testes manuais.

5. Observações adicionais

- Essa melhoria garante que todas as regras de negócio sejam aplicadas no backend, evitando inconsistências entre testes e produção.
- Também facilita futuras integrações e manutenção de automação de testes.