



BARROCO
CONSULT

DIAGNÓSTICO DE QUALIDADE

Relatório de diagnóstico

Restful-Booker (API pública de demonstração)

Relatório demonstrativo

POST /auth	200	{"token":"..."}
POST /booking	200	(grava data "0NaN-aN-aN")
POST /booking	500	(campos ausentes)
DELETE /booking/{id}	201	(deveria ser 200/204)

18 de outubro de 2026
Data

12
Endpoints testados

API
Dispositivos

Testes de contrato
API pública

Resumo executivo

Testamos o contrato da API de reservas: códigos de status, validação de entrada, autenticação, consistência e tempo de resposta a baixa taxa (1 usuário virtual). O tempo de resposta é excelente, mas há falhas sérias de validação: datas inválidas são gravadas, campos obrigatórios ausentes derrubam o servidor e credenciais erradas respondem 200.

Achados por severidade



Principais riscos

- 1 Data inválida é aceita e gravada como "ONaN-aN-aN"** **ALTA**
Reservas ficam com datas corrompidas, quebrando relatórios e integrações.
- 2 Campos obrigatórios ausentes derrubam o servidor (500)** **ALTA**
Entrada malformada vira erro de servidor e pode mascarar instabilidade real.
- 3 Credenciais inválidas respondem 200** **ALTA**
Integrações não distinguem sucesso de falha pelo status, abrindo espaço para bugs de segurança.
- 4 Preço em texto é aceito e vira null** **ALTA**
Reservas perdem o valor silenciosamente, afetando cobrança e relatórios.
- 5 Filtro de datas diverge da documentação** **MÉDIA**
Consumidores do filtro perdem registros e tomam decisões com dados incompletos.

O que já está bom

- Tempo de resposta excelente: p95 de 11 ms com 1 usuário virtual
- Recurso inexistente responde 404 corretamente
- JSON malformado é rejeitado com 400
- Token é emitido para credenciais válidas e exigido em PUT/DELETE
- Criar, ler, atualizar e excluir no caminho feliz funcionam

Medições

k6 · 1 VU · 60 requisições

Métrica	API	Bom até
p50	9 ms	-
p95	11 ms	300 ms
Máximo	19 ms	800 ms

Medição de laboratório com 1 usuário virtual (k6); não representa carga real.

Achados por categoria

Autenticação e segurança

3

Credenciais inválidas respondem 200

ALTA

Login com usuário/senha errados ou vazios retorna HTTP 200 com {"reason":"Bad credentials"}, em vez de 401.

Impacto no negócio: Integrações não distinguem sucesso de falha pelo status, abrindo espaço para bugs de segurança.

Recomendação: Responder 401 Unauthorized quando a autenticação falhar.

```
POST /auth
{"username":"errado","password":"errado"}

HTTP 200

{"reason":"Bad credentials"}
```

API-AUTH-BADCREDS-200 · Onde: POST /auth · Ocorrências: 2 · Esforço: Baixo

Cabeçalhos de segurança ausentes

MÉDIA

As respostas não trazem Strict-Transport-Security nem Content-Security-Policy.

Impacto no negócio: Aumenta a exposição a interceptação e a conteúdo malicioso.

Recomendação: Adicionar HSTS e uma política de conteúdo adequada na camada de API/gateway.

```
GET /booking/745

HTTP 200
server: Cloudflare
via: 1.1 vegur (Heroku)
content-type: application/json; charset=utf-8
x-powered-by: Express
strict-transport-security: (ausente)
content-security-policy: (ausente)
```

API-SEC-HEADERS · Onde: todas as respostas · Ocorrências: 1 · Esforço: Baixo

Tecnologia do servidor exposta

BAIXA

O cabeçalho X-Powered-By revela a stack (Express).

Impacto no negócio: Facilita o mapeamento de vulnerabilidades por atacantes.

Recomendação: Remover o cabeçalho X-Powered-By.

```
GET /booking/745

HTTP 200
x-powered-by: Express
server: Cloudflare
via: 1.1 vegur (Heroku)
```

API-VERSION-EXPOSED · Onde: todas as respostas · Ocorrências: 1 · Esforço: Baixo

Contrato, validação e status codes

8

Data inválida é aceita e gravada como "0NaN-aN-aN"

ALTA

Ao enviar datas em texto, a API responde 200 e persiste "0NaN-aN-aN" nos campos de data, em vez de rejeitar a requisição.

Impacto no negócio: Reservas ficam com datas corrompidas, quebrando relatórios e integrações.

Recomendação: Validar o formato das datas no servidor e responder 400 quando forem inválidas.

```
POST /booking
{"checkin":"nao-e-data","checkout":"tambem-nao"}

HTTP 200

{"bookingdates":{"checkin":"0NaN-aN-aN","checkout":"0NaN-aN-aN"}}
```

API-DATE-STORED-NAN · Onde: POST /booking · Ocorrências: 1 · Esforço: Médio

Campos obrigatórios ausentes derrubam o servidor (500)

ALTA

Uma criação sem os campos obrigatórios responde HTTP 500 Internal Server Error, em vez de 400 com a validação.

Impacto no negócio: Entrada malformada vira erro de servidor e pode mascarar instabilidade real.

Recomendação: Validar o corpo da requisição e responder 400 com a lista de campos faltantes.

```
POST /booking
{"firstname":"Teste"}

HTTP 500

Internal Server Error
```

API-MISSING-FIELDS-500 · Onde: POST /booking · Ocorrências: 1 · Esforço: Baixo

Preço em texto é aceito e vira null

ALTA

Enviar o preço como texto responde 200 e grava totalprice como null, sem avisar o cliente.

Impacto no negócio: Reservas perdem o valor silenciosamente, afetando cobrança e relatórios.

Recomendação: Validar o tipo numérico e rejeitar valores inválidos com 400.

```
POST /booking
{"totalprice":"cem"}

HTTP 200

{"booking":{"totalprice":null}}
```

API-PRICE-TYPE-NUL · Onde: POST /booking · Ocorrências: 1 · Esforço: Baixo

Preço negativo é aceito

MÉDIA

Uma reserva com totalprice -500 é criada normalmente (HTTP 200).

Impacto no negócio: Permite valores incoerentes que distorcem faturamento.

Recomendação: Rejeitar valores negativos com 400.

```
POST /booking
{"totalprice":-500}

HTTP 200

{"booking":{"totalprice":-500}}
```

API-NEGATIVE-PRICE · Onde: POST /booking · Ocorrências: 1 · Esforço: Baixo

Check-out antes do check-in é aceito

MÉDIA

A API cria reservas com a data de saída anterior à de entrada (HTTP 200).

Impacto no negócio: Gera reservas logicamente impossíveis.

Recomendação: Validar a ordem das datas e rejeitar com 400.

```
POST /booking
{"checkin":"2026-11-10","checkout":"2026-11-05"}

HTTP 200

{"bookingdates":{"checkin":"2026-11-10","checkout":"2026-11-05"}}
```

API-DATE-ORDER · Onde: POST /booking · Ocorrências: 1 · Esforço: Baixo

Nome vazio é aceito

MÉDIA

É possível criar uma reserva com firstname em branco (HTTP 200).

Impacto no negócio: Reservas sem identificação dificultam atendimento e relatórios.

Recomendação: Exigir campos de texto não vazios e responder 400 quando ausentes.

```
POST /booking
{"firstname":""}

HTTP 200

{"booking":{"firstname":""}}
```

API-EMPTY-NAME · Onde: POST /booking · Ocorrências: 1 · Esforço: Baixo

Filtro de datas diverge da documentação

MÉDIA

A documentação diz que o filtro de check-in traz datas maiores ou iguais, mas a reserva com a data exata é excluída, comportando-se como "estritamente maior".

Impacto no negócio: Consumidores do filtro perdem registros e tomam decisões com dados incompletos.

Recomendação: Alinhar o comportamento do filtro ao contrato documentado e cobrir com testes.

API-FILTER-BOUNDARY · Onde: GET /booking?checkin=/checkout= · Ocorrências: 2 · Esforço: Médio · Evidência: checkin=2027-03-15 exclui a reserva com check-in 2027-03-15; checkin=2027-03-14 a inclui

Status codes incoerentes em casos de borda

BAIXA

Exclusão responde 201 Created; repetir a exclusão responde 405; PUT na coleção responde 404; um tipo de resposta não suportado responde 418.

Impacto no negócio: Clientes da API tratam erros de forma inconsistente.

Recomendação: Padronizar os status: 200/204 na exclusão, 404 em recurso inexistente, 405 em método não permitido, 406 em tipo não suportado.

API-STATUS-CODES · Onde: DELETE/PUT /booking · Ocorrências: 4 · Esforço: Baixo · Evidência: DELETE ok → 201; DELETE repetido → 405; PUT /booking → 404; Accept: text/csv → 418

Plano de 30 dias

Semana 1

Corrigir o que bloqueia uso e vendas

- Data inválida é aceita e gravada como "ONaN-aN-aN"
- Campos obrigatórios ausentes derrubam o servidor (500)
- Credenciais inválidas respondem 200

Semana 2

Fechar os riscos altos restantes

- Preço em texto é aceito e vira null

Semana 3

Ajustes de qualidade

- Preço negativo é aceito
- Check-out antes do check-in é aceito
- Nome vazio é aceito
- Filtro de datas diverge da documentação
- Cabeçalhos de segurança ausentes

Semana 4

Prevenir regressões

- Suíte de testes de contrato no pipeline
- Validação de entrada e datas no servidor
- Padronização dos status codes

Limitações deste diagnóstico

- Testes feitos na API pública de demonstração, criada com falhas intencionais, a baixa taxa e sem carga
- As reservas de teste criadas foram removidas ao final
- Tempo de resposta medido em laboratório com 1 usuário virtual; não representa carga real

Checagens não executadas

- Teste de carga: fora do escopo; medimos apenas tempo de resposta a baixa taxa

Próximos passos

- Rodar a suíte de contrato a cada publicação
- Cobrir validação de entrada e autenticação com testes automatizados

Pacote sugerido

80h

horas por mês

API com várias falhas de contrato e validação, incluindo dado inválido persistido e erro 500 em entrada malformada.

Cobertura contínua dos endpoints com testes de contrato e validação, mais regressão.

Valores sob consulta.

Vamos resolver esses pontos juntos?

Agende uma conversa de 30 minutos com a equipe da Barroco: explicamos cada achado e montamos o plano para o seu produto.

barrocoqa.com/pt · contato@barrocoqa.com

Agendar conversa

Diagnóstico gerado com apoio de IA e validado por um analista humano. Relatório demonstrativo, de divulgação pública.

