



BARROCO
CONSULT

QUALITY DIAGNOSIS

Diagnosis report

Restful-Booker (public demonstration API)

Demonstration report

POST /auth	200	{"token":"..."}
POST /booking	200	(stores date "0NaN-aN-aN")
POST /booking	500	(missing fields)
DELETE /booking/{id}	201	(should be 200/204)

October 18, 2026
Date

12
Endpoints tested

API
Devices

Contract testing
public API

Executive summary

We tested the booking API contract: status codes, input validation, authentication, consistency and response time at a low rate (1 virtual user). Response time is excellent, but there are serious validation failures: invalid dates are stored, missing required fields crash the server, and wrong credentials respond 200.

Findings by severity



Top risks

- 1 Invalid date is accepted and stored as "ONaN-aN-aN" HIGH**
Bookings end up with corrupted dates, breaking reports and integrations.
- 2 Missing required fields crash the server (500) HIGH**
Malformed input becomes a server error and can mask real instability.
- 3 Invalid credentials respond 200 HIGH**
Integrations cannot tell success from failure by status, enabling security bugs.
- 4 Text price is accepted and becomes null HIGH**
Bookings silently lose their amount, affecting billing and reports.
- 5 Date filter diverges from the documentation MEDIUM**
Filter consumers miss records and decide on incomplete data.

What is already good

- Excellent response time: p95 of 11 ms with 1 virtual user
- Missing resource correctly responds 404
- Malformed JSON is rejected with 400
- A token is issued for valid credentials and required on PUT/DELETE
- Create, read, update and delete work on the happy path

Measurements

k6 · 1 VU · 60 requests

Metric	API	Good up to
p50	9 ms	-
p95	11 ms	300 ms
Max	19 ms	800 ms

Lab measurement with 1 virtual user (k6); it does not represent real load.

Findings by category

Authentication and security

3

Invalid credentials respond 200

HIGH

Login with wrong or empty username/password returns HTTP 200 with {"reason":"Bad credentials"} instead of 401.

Business impact: Integrations cannot tell success from failure by status, enabling security bugs.

Recommendation: Respond 401 Unauthorized when authentication fails.

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

HTTP 200

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

API-AUTH-BADCREDS-200 · Where: POST /auth · Occurrences: 2 · Effort: Low

Security headers missing

MEDIUM

Responses carry neither Strict-Transport-Security nor Content-Security-Policy.

Business impact: Raises exposure to interception and malicious content.

Recommendation: Add HSTS and a suitable content policy at the API/gateway layer.

```
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: (absent)
content-security-policy: (absent)
```

API-SEC-HEADERS · Where: all responses · Occurrences: 1 · Effort: Low

Server technology exposed

LOW

The X-Powered-By header reveals the stack (Express).

Business impact: Makes vulnerability mapping easier for attackers.

Recommendation: Remove the X-Powered-By header.

```
GET /booking/745

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

API-VERSION-EXPOSED · Where: all responses · Occurrences: 1 · Effort: Low

Invalid date is accepted and stored as "0NaN-aN-aN"

HIGH

Sending dates as text returns 200 and persists "0NaN-aN-aN" in the date fields instead of rejecting the request.

Business impact: Bookings end up with corrupted dates, breaking reports and integrations.

Recommendation: Validate the date format on the server and respond 400 when invalid.

```
POST /booking
{"checkin":"not-a-date","checkout":"also-not"}

HTTP 200

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

API-DATE-STORED-NAN · Where: POST /booking · Occurrences: 1 · Effort: Medium

Missing required fields crash the server (500)

HIGH

Creating without the required fields returns HTTP 500 Internal Server Error instead of 400 with validation.

Business impact: Malformed input becomes a server error and can mask real instability.

Recommendation: Validate the request body and respond 400 with the list of missing fields.

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

HTTP 500

Internal Server Error
```

API-MISSING-FIELDS-500 · Where: POST /booking · Occurrences: 1 · Effort: Low

Text price is accepted and becomes null

HIGH

Sending the price as text returns 200 and stores totalprice as null, with no warning to the client.

Business impact: Bookings silently lose their amount, affecting billing and reports.

Recommendation: Validate the numeric type and reject invalid values with 400.

```
POST /booking
{"totalprice":"one hundred"}

HTTP 200

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

API-PRICE-TYPE-NULL · Where: POST /booking · Occurrences: 1 · Effort: Low

Negative price is accepted

MEDIUM

A booking with totalprice -500 is created normally (HTTP 200).

Business impact: Allows incoherent values that distort billing.

Recommendation: Reject negative values with 400.

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

HTTP 200

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

API-NEGATIVE-PRICE · Where: POST /booking · Occurrences: 1 · Effort: Low

Check-out before check-in is accepted

MEDIUM

The API creates bookings with the checkout date before the checkin date (HTTP 200).

Business impact: Produces logically impossible bookings.

Recommendation: Validate date order and reject with 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 · Where: POST /booking · Occurrences: 1 · Effort: Low

Empty name is accepted

MEDIUM

A booking can be created with a blank firstname (HTTP 200).

Business impact: Unidentified bookings hamper support and reporting.

Recommendation: Require non-empty text fields and respond 400 when absent.

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

HTTP 200

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

API-EMPTY-NAME · Where: POST /booking · Occurrences: 1 · Effort: Low

Date filter diverges from the documentation

MEDIUM

The docs say the check-in filter returns dates greater than or equal, but the booking with the exact date is excluded, behaving as "strictly greater".

Business impact: Filter consumers miss records and decide on incomplete data.

Recommendation: Align the filter behavior with the documented contract and cover it with tests.

API-FILTER-BOUNDARY · Where: GET /booking?checkin=/checkout= · Occurrences: 2 · Effort: Medium · Evidence: checkin=2027-03-15 excludes the booking with check-in 2027-03-15; checkin=2027-03-14 includes it

Inconsistent status codes on edge cases

LOW

Deletion responds 201 Created; repeating the deletion responds 405; PUT on the collection responds 404; an unsupported response type responds 418.

Business impact: API clients handle errors inconsistently.

Recommendation: Standardize statuses: 200/204 on delete, 404 on missing resource, 405 on method not allowed, 406 on unsupported type.

API-STATUS-CODES · Where: DELETE/PUT /booking · Occurrences: 4 · Effort: Low · Evidence: DELETE ok → 201; repeated DELETE → 405; PUT /booking → 404; Accept: text/csv → 418

30-day plan

Week 1 Fix what blocks usage and sales - Invalid date is accepted and stored as "0NaN-aN-aN" - Missing required fields crash the server (500) - Invalid credentials respond 200	Week 2 Close the remaining high risks Text price is accepted and becomes null	Week 3 Quality improvements - Negative price is accepted - Check-out before check-in is accepted - Empty name is accepted - Date filter diverges from the documentation - Security headers missing	Week 4 Prevent regressions - Contract test suite in the pipeline - Server-side input and date validation - Standardized status codes
---	---	--	--

Limitations of this diagnosis

- Tests run on the public demo API, built with intentional bugs, at a low rate and without load
- The test bookings created were removed at the end
- Response time measured in a lab with 1 virtual user; it does not represent real load

Checks not run

- Load testing: out of scope; we only measured response time at a low rate

Next steps

- Run the contract suite on every release
- Cover input validation and authentication with automated tests

Suggested package

80h

hours per month

An API with several contract and validation failures, including invalid data persisted and a 500 error on malformed input.

Continuous endpoint coverage with contract and validation tests, plus regression.

Pricing on request.

Shall we fix these together?

Book a 30-minute call with the Barroco team: we walk you through every finding and build the plan for your product.

[Book a call](#)

barrocoqa.com · contactus@barrocoqa.com

Diagnostic produced with AI assistance and validated by a human analyst. Sample report, for public distribution.

