Performance Test Plan

Performance Test Plan

> **Project:** Bilko **Version:** 1.0 **Date:** 2026-02-25 **Author:** Ops Architect **Status:** Final **Reviewers:** Tech Lead, Alem Bašić

Document History

Version	Date	Author	Changes
0.1	2026-02-23	Ops Architect	Initial draft
1.0	2026-02-25	ALAI Documentation Team	Finalized — approved for production use

---

1. Overview

Status: PLANNED — Phase 2 (post-MVP)

Performance testing for Bilko covers API load testing and frontend Core Web Vitals. At MVP stage (< 500 users), Railway Starter (2GB RAM) is expected to handle load without dedicated performance testing. Formal performance testing will be run before scaling to 1,000+ users.

Tool: k6 (API load testing) + Lighthouse CI (frontend) **Target environment:** Staging (production-sized data) — NEVER run load tests on production **Responsible:** DevOps (Ops Architect at MVP stage)

---

2. Performance Requirements

API Performance Targets

Metric	MVP Target (10 concurrent)	Scale Target (100 concurrent)
P50 response time	< 100ms	< 200ms
P95 response time	< 500ms	< 1000ms
P99 response time	< 1000ms	< 2000ms
Throughput	> 50 req/s	> 500 req/s
Error rate under load	< 0.5%	< 1%
Max memory (Railway)	< 1.5GB	< 7GB (Pro plan)

Frontend Performance Targets (Core Web Vitals)

Metric	Target	Tool
LCP (Largest Contentful Paint)	< 2.5s	Lighthouse CI / Vercel Analytics
FID / INP	< 200ms	Lighthouse CI / Vercel Analytics
CLS (Cumulative Layout Shift)	< 0.1	Lighthouse CI / Vercel Analytics
TTFB (Time to First Byte)	< 800ms	Lighthouse CI
Performance Score	> 90	Lighthouse CI

Database Performance Targets

Query Type	Target P95
Invoice list (paginated, 20 items)	< 50ms
Invoice create (with items)	< 100ms
VAT report generation (monthly)	< 2000ms
Balance sheet calculation	< 3000ms
Double-entry transaction insert	< 50ms

---

3. Load Test Scenarios

Scenario 1: Invoice Creation Under Load (MVP Critical Path)

Simulates: 10 concurrent users creating invoices simultaneously

```
// apps/e2e/load/invoice-create.js
import http from 'k6/http'
import { check, sleep } from 'k6'
import { Rate } from 'k6/metrics'

const errorRate = new Rate('errors')

export let options = {
  vus: 10, // 10 virtual users (MVP target)
  duration: '2m', // 2-minute steady-state load
  thresholds: {
    http_req_duration: ['p(95)<500'], // P95 < 500ms
    http_req_failed: ['rate<0.01'], // < 1% error rate
    errors: ['rate<0.01'],
  },
}

const BASE_URL = __ENV.BASE_URL || 'https://staging-api.bilko.io'

export default function () {
  // Login
  const loginRes = http.post(
    `${BASE_URL}/api/v1/auth/login`,
    JSON.stringify({
      email: `loadtest_${__VU}@bilko.io`,
      password: 'LoadTest123!',
    }),
    { headers: { 'Content-Type': 'application/json' } },
  )

  check(loginRes, { 'login success': (r) => r.status === 200 })
  const token = loginRes.json('accessToken')

  // Create invoice
  const invoiceRes = http.post(
    `${BASE_URL}/api/v1/invoices`,
    JSON.stringify({
      customerId: __ENV.TEST_CUSTOMER_ID,
```

```
      invoiceDate: '2026-02-23',
      dueDate: '2026-03-23',
      currencyCode: 'RSD',
      items: [{ description: 'Load test item', quantity: 1, unitPrice: '5000.0000', taxRate:
20 }],
    }),
    {
      headers: {
        'Content-Type': 'application/json',
        Authorization: `Bearer ${token}`,
      },
    },
  )

  const success = check(invoiceRes, {
    'invoice created': (r) => r.status === 201,
    'has invoice number': (r) => r.json('invoiceNumber') !== undefined,
    'total correct': (r) => r.json('totalAmount') === '6000.0000',
  })

  errorRate.add(!success)
  sleep(1)
}
```

Scenario 2: Dashboard Load (Read-Heavy)

Simulates: 50 concurrent users viewing dashboard

```
// apps/e2e/load/dashboard.js
export let options = {
  vus: 50,
  duration: '5m',
  thresholds: {
    http_req_duration: ['p(95)<1000'],
    http_req_failed: ['rate<0.01'],
  },
}

export default function () {
```

```
  const headers = { Authorization: `Bearer ${__ENV.AUTH_TOKEN}` }

  http.get(`${BASE_URL}/api/v1/invoices?limit=20`, { headers })
  http.get(`${BASE_URL}/api/v1/expenses?limit=20`, { headers })
  http.get(`${BASE_URL}/api/v1/reports/dashboard-summary`, { headers })

  sleep(Math.random() * 3 + 1)
}
```

Scenario 3: VAT Report Generation (Heavy Query)

Simulates: 10 concurrent accountants generating VAT reports (monthly)

```
// apps/e2e/load/vat-report.js
export let options = {
  vus: 10,
  duration: '1m',
  thresholds: {
    http_req_duration: ['p(95)<3000'], // VAT reports can take up to 3s
    http_req_failed: ['rate<0.01'],
  },
}

export default function () {
  const res = http.get(`${BASE_URL}/api/v1/reports/vat?startDate=2026-01-01&endDate=2026-01-
31`, {
    headers: { Authorization: `Bearer ${__ENV.AUTH_TOKEN}` },
  })
  check(res, { 'vat report success': (r) => r.status === 200 })
  sleep(5) // Accountants don't generate reports rapidly
}
```

---

4. Lighthouse CI Configuration (PLANNED)

```
// lighthouserc.js
module.exports = {
  ci: {
    collect: {
      url: [
        'https://staging.bilko.io/',
        'https://staging.bilko.io/invoices',
        'https://staging.bilko.io/dashboard',
      ],
      numberOfRuns: 3,
    },
    assert: {
      assertions: {
        'categories:performance': ['error', { minScore: 0.9 }],
        'categories:accessibility': ['error', { minScore: 0.9 }],
        'first-contentful-paint': ['error', { maxNumericValue: 2000 }],
        'largest-contentful-paint': ['error', { maxNumericValue: 2500 }],
        'cumulative-layout-shift': ['error', { maxNumericValue: 0.1 }],
      },
    },
    upload: {
      target: 'temporary-public-storage',
    },
  },
}
```

---

5. Database Query Performance Tests

Critical queries to benchmark before production:

```
-- Test 1: Invoice list query performance (target: < 50ms)
EXPLAIN ANALYZE
SELECT i.*, c.name as customer_name
FROM invoices i
JOIN contacts c ON c.id = i."customerId"
WHERE i."organizationId" = $1
  AND i."deletedAt" IS NULL
ORDER BY i."createdAt" DESC
```

```
LIMIT 20 OFFSET 0;

-- Test 2: VAT report aggregation (target: < 2000ms)
EXPLAIN ANALYZE
SELECT
  SUM(total_amount) as total_invoiced,
  SUM(tax_amount) as total_vat_collected
FROM invoices
WHERE "organizationId" = $1
  AND invoice_date BETWEEN '2026-01-01' AND '2026-01-31'
  AND status IN ('sent', 'paid');

-- Test 3: Double-entry balance check (target: < 100ms)
EXPLAIN ANALYZE
SELECT account_id, SUM(CASE WHEN type = 'debit' THEN amount ELSE -amount END) as balance
FROM transactions
WHERE "organizationId" = $1
GROUP BY account_id
HAVING ABS(SUM(CASE WHEN type = 'debit' THEN amount ELSE -amount END)) > 0.0001;
```

Required indexes (verify these exist post-migration):

- `invoices(organizationId, deletedAt, createdAt)` — invoice list query
- `transactions(organizationId, account_id)` — balance check
- `invoices(organizationId, invoiceDate, status)` — VAT report

---

6. Performance Test Execution

```
# Run load test (requires k6 installed)
k6 run apps/e2e/load/invoice-create.js \
  -e BASE_URL=https://staging-api.bilko.io \
  -e TEST_CUSTOMER_ID=<uuid>

# Run with increased VUs for scale testing
k6 run --vus 100 --duration 5m apps/e2e/load/dashboard.js

# Run Lighthouse CI (requires lhci installed)
lhci autorun
```

```
# Database query benchmarking
railway run psql $DATABASE_URL -f apps/e2e/load/benchmark-queries.sql
```

7. Performance Baseline & Regression

Before each major release:

1. Run all load test scenarios on staging
2. Record results in performance baseline table
3. Compare to previous baseline — alert if P95 degraded > 20%
4. Identify and fix regressions before deploying to production

Release	Date	Inv Create P95	Dashboard P95	VAT Report P95
MVP baseline	Not yet measured (pre-launch)	Not yet measured	Not yet measured	Not yet measured

Related Documents

- Test Strategy
- Monitoring & Observability
- Deployment Architecture

Approval

Role	Name	Date	Signature
Author	Ops Architect	2026-02-23	
Reviewer	Tech Lead		
Approver	Alem Bašić		

Revision #9
Created 2026-02-24 22:50:56 UTC by John
Updated 2026-06-21 20:02:25 UTC by John