Operations

Bilko operational runbooks, incident management, and SLA documentation

- Go-Live Runbook
- Incident Report Template
- Operational Runbook
- Post-Mortem Template
- SLA Report

Go-Live Runbook

Go-Live Runbook

> **Project:** Bilko **Version:** 0.1 — Initial Production Launch **Date:** 2026-02-23
> **Author:** Ops Architect **Status:** Draft **Reviewers:** Tech Lead, Alem Bašić

Document History

Version	Date	Author	Changes
0.1	2026-02-23	Ops Architect	Initial draft

---

1. Go-Live Overview

What: Bilko v1.0 — first production launch of the cloud accounting SaaS **Target date:** TBD (set when MVP backend complete) **Deployment window:** 2-hour window (prefer low-traffic: Tuesday–Thursday 10:00–12:00 CET) **Go-Live Type:** New product launch — no existing users to migrate

Incident Commander: Alem Bašić (+47 40 47 42 51) — primary **Technical Lead:** Alem Bašić (also, at MVP) **War Room:** Slack #bilko-deploys (create dedicated #bilko-launch channel for day-of) **Status Page:** status.bilko.io (PLANNED — configure BetterStack before launch)

---

2. Pre-Launch Checklist

T-7 Days: Infrastructure Verification

- ☐ Railway project created with `api` service and PostgreSQL 15 database
- ☐ Railway EU West region confirmed (GDPR compliance)
- ☐ Vercel project created and linked to GitHub repo (`alai-holding/bilko`)

- [ ] Cloudflare R2 bucket `bilko-receipts` created with correct CORS policy
- [ ] All production environment variables set in Railway and Vercel dashboards
- [ ] Railway health check endpoint (`GET /health`) responding `{"status":"ok","db":"ok"}`
- [ ] Vercel build successful (`pnpm run build` on main branch)
- [ ] Database migrations applied: `railway run npx prisma migrate deploy`
- [ ] Cost estimate confirmed within budget (< €25/mo at MVP)

Owner: Alem Bašić | **Due:** T-7 days

T-5 Days: DNS Configuration

- [ ] Cloudflare DNS records created for bilko.io:
  - `@` CNAME → `cname.vercel-dns.com` (Proxied: Yes)
  - `www` CNAME → `cname.vercel-dns.com` (Proxied: Yes)
  - `api` CNAME → `<railway-domain>.railway.app` (Proxied: No)
- [ ] bilko.io domain verified in Vercel project (Vercel → Settings → Domains)
- [ ] api.bilko.io custom domain set in Railway (Railway → Settings → Domains)
- [ ] DNS propagation verified: `dig bilko.io` and `dig api.bilko.io`
- [ ] Cloudflare "Always Use HTTPS" enabled for bilko.io
- [ ] bilko.rs redirect rule configured (if domain registered): bilko.rs → bilko.io

Owner: Alem Bašić | **Due:** T-5 days

T-5 Days: SSL Certificates

- [ ] bilko.io TLS certificate provisioned (Vercel auto-provisions via Let's Encrypt)
- [ ] api.bilko.io TLS certificate provisioned (Railway auto-provisions)
- [ ] HTTPS verified: `curl -I https://bilko.io` → HTTP/2 200
- [ ] HTTPS verified: `curl https://api.bilko.io/health` → `{"status":"ok"}`
- [ ] HTTP → HTTPS redirect working: `curl -I http://bilko.io` → 301

Owner: Alem Bašić | **Due:** T-5 days

T-3 Days: Third-Party Integrations

- [ ] SendGrid live API key in Railway production secrets
- [ ] SendGrid domain authentication for bilko.io completed (SPF, DKIM, DMARC)
- [ ] Test email sent from `noreply@bilko.io` and received successfully
- [ ] Cloudflare R2 live API credentials in Railway production secrets
- [ ] Test file upload to R2 `bilko-receipts` bucket via API

Owner: Alem Bašić | **Due:** T-3 days

T-2 Days: Monitoring Setup

- [ ] Sentry project created for Bilko backend + frontend (if ready)
- [ ] SENTRY_DSN environment variable set in Railway and Vercel
- [ ] BetterStack uptime monitors created:
  - `https://bilko.io` — check every 1 min
  - `https://api.bilko.io/health` — check every 1 min
- [ ] BetterStack alert routing: Slack #bilko-alerts + email to alem@alai.no
- [ ] Alert test fired and received by Alem Bašić

Owner: Alem Bašić | **Due:** T-2 days

T-1 Day: Legal / Compliance

- [ ] Privacy policy published at bilko.io/privacy
- [ ] Terms of service published at bilko.io/terms
- [ ] Cookie consent banner implemented (not required for SaaS with no tracking cookies, but review)
- [ ] GDPR data processing documentation completed
- [ ] Data retention policy documented (financial records: 10 years per Serbian law)
- [ ] Legal sign-off from Alem Bašić on compliance readiness

Owner: Alem Bašić | **Due:** T-1 day

T-1 Day: Backup Verification

- [ ] Railway automated backup confirmed running (PostgreSQL → Backups tab)

☐ Manual backup taken and restore tested on staging:

```
railway run pg_dump $DATABASE_URL -f test_backup.dump
# Restore to staging DB
railway run psql $STAGING_DATABASE_URL < test_backup.dump
```

☐ Backup verified: record counts match original

Owner: Alem Bašić | **Due:** T-1 day

T-0: Final Checks (1 hour before launch)

☐ Staging smoke tests all green (last run: within 24h)
☐ Main branch is up to date with all intended changes
☐ Railway and Vercel dashboards open in browser
☐ BetterStack monitoring dashboard open
☐ Rollback procedure reviewed (< 2 min for frontend, < 5 min for backend)
☐ Slack #bilko-launch channel open with Alem

---

3. Launch Day Procedure

H+0:00 — Deployment Start

Time	Action	Owner	Status
H+0:00	Post in #bilko-launch: "Bilko launch starting"	Alem	
H+0:00	Confirm Railway deployment pipeline ready	Alem	
H+0:05	Trigger production deployment from main branch	Alem	
H+0:10	Monitor Railway build logs	Alem	

H+0:10 ? H+0:20 — Deploy Verification

Time	Action	Owner	Status

H+0:15	Confirm Railway deployment successful	Alem	
H+0:15	Verify health check: `curl https://api.bilko.io/health`	Alem	
H+0:20	Confirm Vercel frontend deployment successful	Alem	
H+0:20	Verify frontend: open https://bilko.io in browser	Alem	

H+0:20 ? H+0:45 — Smoke Tests

Time	Action	Owner	Status
H+0:20	Register new test account on bilko.io	Alem	
H+0:25	Create test invoice (RSD, 20% VAT)	Alem	
H+0:30	Verify invoice totals (subtotal + VAT = total)	Alem	
H+0:35	Create test expense with receipt upload	Alem	
H+0:40	Generate VAT report for current month	Alem	
H+0:45	All smoke tests PASS → proceed	Alem	

H+0:45 — Go-Live Declaration

Time	Action	Owner	Status
H+0:45	Post in #bilko-launch: "Bilko is LIVE! bilko.io"	Alem	
H+0:50	Update status page: "All systems operational"	Alem	
H+1:00	Send launch announcement (if planned)	Alem	

---

4. Post-Launch Monitoring (First 48 Hours)

Period	Check Frequency	What to Watch
H+0 to H+4	Every 30 min	BetterStack dashboard, Railway metrics, Sentry errors
H+4 to H+24	Every 60 min	Same as above
Day 2	Every 4 hours	Same as above
Day 3+	Standard monitoring	BetterStack alerts only

Healthy indicators:

- BetterStack: all monitors green
- Railway CPU: < 50%, Memory: < 1GB
- Sentry: 0 new issues in first hour
- API health endpoint: `{"status":"ok","db":"ok"}`

---

5. Rollback Triggers & Procedure

Rollback if:

- Health check fails for > 3 consecutive minutes
- Error rate > 5% in any 5-minute window (Sentry)
- Financial calculation bug discovered (any VAT/total error)
- Authentication completely broken

Rollback procedure:

1. Post in #bilko-launch: "Rolling back — [reason]"
2. **Frontend:** Vercel Dashboard → Deployments → Promote previous → instant
3. **Backend:** Railway Dashboard → Deployments → Redeploy previous → ~2 min
4. Verify health: `curl https://api.bilko.io/health`
5. Post update when rollback complete

---

6. Communication Plan

Launch Day Communications

Audience	Channel	When	Message
Internal	Slack #bilko-launch	H+0	"Deployment started"
Internal	Slack #bilko-launch	H+0:45	"Bilko is live! bilko.io"

Audience	Channel	When	Message
Beta users (if any)	Email	H+1:00	Launch announcement
Status page	status.bilko.io	H+0:45	"All systems operational"

Related Documents

- Deployment Checklist
- Rollback Plan
- Operational Runbook
- Monitoring & Observability
- Disaster Recovery Plan

Approval

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

Incident Report Template

Incident Report

> **Project:** Bilko **Version:** 0.1 **Date:** 2026-02-23 **Author:** Ops Architect **Status:** Draft (Template — fill in per incident) **Reviewers:** Tech Lead, Alem Bašić

Document History

Version	Date	Author	Changes
0.1	2026-02-23	Ops Architect	Initial draft

INSTRUCTIONS

Create a new incident report file for each incident:

- Filename: `INCIDENT-YYYY-MM-DD-<short-title>.md`
- Location: `docs/operations/incidents/`
- Fill in all sections within 48 hours of incident resolution
- P0 incidents require a full post-mortem (see `post-mortem.md`)

Incident Report: [SHORT TITLE]

Incident ID: INC-YYYY-MM-DD-NNN **Reported by:** [Name] **Date detected:** YYYY-MM-DD HH:MM CET **Date resolved:** YYYY-MM-DD HH:MM CET **Total duration:** X hours Y minutes **Severity:** P0 / P1 / P2 / P3

---

1. Incident Summary

[Example: On YYYY-MM-DD at HH:MM CET, the Bilko API became unavailable due to an out-of-memory error on Railway. All users were unable to create invoices or access their dashboard for 47 minutes. The incident was resolved by restarting the Railway service and deploying a memory optimization patch.]

---

2. Impact

Metric	Value
Duration	X min
Users affected	[All / Specific org / None]
Data loss	[None / Describe if any]
Financial records at risk	[None / Describe if any]
Revenue impact	[None / Describe if applicable]
GDPR reportable	[Yes / No] — if Yes, notify Datatilsynet within 72h

---

3. Timeline

Time (CET)	Event
HH:MM	First signs of degradation (detected by: BetterStack / user report / Sentry)
HH:MM	Incident declared by [Name]
HH:MM	[First diagnosis step]
HH:MM	[Root cause identified]
HH:MM	[Fix applied]
HH:MM	Service restored, health check green
HH:MM	Incident resolved, monitoring period started
HH:MM	Monitoring period ended — all clear

---

4. Root Cause Analysis

Root cause: [One sentence — the actual technical cause]

Example root causes for Bilko:

- Railway API service exhausted 2GB RAM limit due to memory leak in invoice PDF generation
- Database connection pool exhausted (25 max connections on Railway Starter) under load
- Bad database migration caused index corruption on `invoices` table
- Expired SendGrid API key (not rotated) caused all invoice emails to fail
- Cloudflare R2 API credentials rotated without updating Railway env vars

Contributing factors:

- [Factor 1: e.g., No memory usage alerting configured]
- [Factor 2: e.g., No automated secret rotation reminder]

---

5. Detection

How was the incident detected?

- ☐ BetterStack uptime monitor alert
- ☐ Sentry error rate spike
- ☐ User report (direct to Alem / support)
- ☐ Routine health check
- ☐ Monitoring dashboard review

Time to detect: [X minutes from first symptom to alert]

Was detection fast enough? [Yes / No — if No, document what would have caught it faster]

---

6. Response

Response Actions Taken

Time	Action	Result
HH:MM	[Action taken]	[Result]

What Worked Well

- [e.g., BetterStack alert fired within 2 min]
- [e.g., Rollback procedure was documented and worked on first try]

What Didn't Work

- [e.g., Railway logs were hard to search for the specific error]
- [e.g., No backup Railway contact — only one person on-call]

---

7. Financial Data Integrity Check

Required for all P0/P1 incidents. Skip for P2/P3 that don't touch accounting.

- ☐ All invoices created during incident window verified (count before vs after)
- ☐ VAT calculations for affected period verified correct
- ☐ Double-entry balances verified for all affected transactions
- ☐ No orphaned transactions (debit without credit) in database
- ☐ Affected organization(s) notified if any data discrepancy found

Verification query (run after incident):

```
-- Check for unbalanced entries during incident window
SELECT t.id, t.created_at, t.amount
FROM transactions t
WHERE t.created_at BETWEEN '<incident_start>' AND '<incident_end>'
  AND t.id NOT IN (
    SELECT DISTINCT "transactionId" FROM transaction_entries WHERE type = 'debit'
  );
-- Expected: 0 rows (all transactions have debit entries)

-- Check invoice totals match line item sums
SELECT i.id, i.total_amount,
       SUM(ii.quantity * ii.unit_price * (1 + ii.tax_rate/100)) as calculated_total
FROM invoices i
JOIN invoice_items ii ON ii."invoiceId" = i.id
WHERE i.created_at BETWEEN '<incident_start>' AND '<incident_end>'
GROUP BY i.id, i.total_amount
HAVING ABS(i.total_amount - SUM(ii.quantity * ii.unit_price * (1 + ii.tax_rate/100))) >
0.0001;
```

```
-- Expected: 0 rows
```

8. User Communication

Channel	When Sent	Content Summary
status.bilko.io	HH:MM	"Investigating service disruption"
status.bilko.io	HH:MM	Status update with ETA
status.bilko.io	HH:MM	"Service restored"
Email to affected users	HH:MM (if needed)	[Summary of impact and resolution]

9. Action Items

#	Action	Owner	Due Date	Priority
1	[Preventive action]	[Owner]	YYYY-MM-DD	P0/P1/P2
2	[Detection improvement]	[Owner]	YYYY-MM-DD	P1
3	[Documentation update]	[Owner]	YYYY-MM-DD	P2

10. Follow-Up

- ☐ Post-mortem scheduled (required for P0 incidents): [Date/Time]
- ☐ Action items added to project backlog
- ☐ Runbook updated with new diagnosis/fix steps
- ☐ Monitoring improved to detect this issue faster next time

Approval

Role	Name	Date	Signature
Author (on-call)			

Role	Name	Date	Signature
Reviewer	Alem Bašić		

Operational Runbook

Operational Runbook

> **Project:** Bilko **Version:** 0.1 **Date:** 2026-02-23 **Author:** Ops Architect **Status:** Draft **Reviewers:** Tech Lead, Alem Bašić

Document History

Version	Date	Author	Changes
0.1	2026-02-23	Ops Architect	Initial draft

---

1. Service Overview

Service	URL	Platform	Health Check
Frontend	https://bilko.io	Vercel	`curl -I https://bilko.io` → 200
API	https://api.bilko.io	Railway EU West	`curl https://api.bilko.io/health`
Database	bilko_prod	Railway PostgreSQL 15	Health check via API
File storage	bilko-receipts	Cloudflare R2	API upload test
Email	noreply@bilko.io	SendGrid	Test email send

On-call: Alem Bašić (+47 40 47 42 51)

---

2. Routine Operations

2.1 Check System Health

```
# API health
curl https://api.bilko.io/health
# Expected: {"status":"ok","db":"ok","timestamp":"..."}

# Frontend
curl -I https://bilko.io
# Expected: HTTP/2 200

# Railway logs (last 50 lines)
railway logs --tail 50

# Railway metrics (via dashboard)
# Railway Dashboard → Project → api → Metrics
```

2.2 Deploy New Version

Standard deploy (automatic):

1. Merge PR to `main` branch
2. GitHub Actions CI pipeline runs automatically
3. On pass: Vercel and Railway auto-deploy
4. Monitor: Railway logs + BetterStack for 15 min post-deploy

Manual deploy (emergency):

```
# Deploy frontend manually
cd apps/web && vercel --prod

# Deploy backend manually
railway up --service api --environment production

# Run migrations before backend deploy
railway run npx prisma migrate deploy
```

2.3 Database Migrations

Never run migrations directly in production without backup:

```
# Step 1: Take backup
railway run pg_dump $DATABASE_URL -f backup_$(date +%Y%m%d_%H%M).dump
```

```
# Step 2: Test migration on staging
railway run --environment staging npx prisma migrate deploy

# Step 3: Apply to production (after staging verification)
railway run npx prisma migrate deploy

# Step 4: Verify
railway run npx prisma db pull  # Confirm schema matches
```

2.4 View Application Logs

```
# Railway API logs (streaming)
railway logs --service api

# Railway API logs (last 100 lines)
railway logs --service api --tail 100

# Filter for errors
railway logs --service api | grep -i error

# Filter for specific organization
railway logs --service api | grep "organizationId=<uuid>"
```

2.5 Environment Variable Updates

```
# View current env vars (Railway)
railway variables list --service api

# Update a secret (Railway CLI)
railway variables set JWT_SECRET=<new-value> --service api --environment production
# Restart service after update
railway service restart --service api

# Vercel env var update
vercel env add NEXT_PUBLIC_API_URL production
```

---

3. Monitoring & Alerting

3.1 Normal Operating Ranges

Metric	Normal Range	Alert if
API CPU	5–30%	> 70% for 5 min
API Memory	200–800MB	> 1.5GB
DB connections	2–10 active	> 20 active
API P95 latency	< 200ms	> 1000ms
Error rate (5xx)	< 0.1%	> 1%
Uptime	100%	Any downtime > 2 min

3.2 BetterStack Dashboards (PLANNED)

- System status: status.bilko.io
- Internal metrics: BetterStack dashboard
- Uptime history: BetterStack → Monitors

3.3 Sentry Error Monitoring (PLANNED)

- Frontend errors: Sentry → bilko-frontend project
- Backend errors: Sentry → bilko-backend project
- Financial logic errors: tagged `financial-logic` in Sentry — P0 response required

---

4. Incident Response Procedures

4.1 API Down (all requests failing)

1. Check Railway Dashboard → api → Status
2. Check health endpoint: `curl https://api.bilko.io/health`
3. Check Railway logs: `railway logs --service api --tail 50`
4. Common causes and fixes:

Cause	Fix
Recent bad deploy	Railway → Deployments → Redeploy previous

Cause	Fix
Out of memory (OOM)	Restart service, investigate memory leak
Database connection exhausted	Restart service, check PgBouncer config
Database down	See 4.2

5. If not resolved in 5 min → post to #bilko-alerts, start incident response

4.2 Database Issues

Connection refused:

```
# Check Railway PostgreSQL status
# Railway Dashboard → Project → PostgreSQL → Status

# Test connection manually
railway run psql $DATABASE_URL -c "SELECT 1;"
```

High connection count:

```
# Check active connections
railway run psql $DATABASE_URL -c "
SELECT count(*), state, wait_event_type
FROM pg_stat_activity
GROUP BY state, wait_event_type
ORDER BY count DESC;"

# Kill idle connections if needed (after investigation)
railway run psql $DATABASE_URL -c "
SELECT pg_terminate_backend(pid)
FROM pg_stat_activity
WHERE state = 'idle'
  AND state_change < NOW() - INTERVAL '10 minutes';"
```

Slow queries:

```
# Find long-running queries
railway run psql $DATABASE_URL -c "
SELECT pid, now() - query_start AS duration, query
FROM pg_stat_activity
WHERE state = 'active' AND query_start < NOW() - INTERVAL '5 seconds'
```

```
ORDER BY duration DESC;"
```

4.3 High Error Rate

1. Open Sentry → bilko-backend → Issues → Sort by date
2. Identify most frequent error in last 15 min
3. Check if error is from recent deploy: Railway → Deployments → check timestamp
4. If financial logic error (VAT/double-entry): **treat as P0, rollback immediately**
5. If non-financial error: assess impact, investigate root cause before rollback

4.4 Storage (R2) Issues

```
# Test R2 connectivity from API
railway run node -e "
const { S3Client, HeadBucketCommand } = require('@aws-sdk/client-s3');
const client = new S3Client({ endpoint: process.env.R2_ENDPOINT, ... });
client.send(new HeadBucketCommand({ Bucket: 'bilko-receipts' }))
  .then(() => console.log('R2 OK'))
  .catch(e => console.error('R2 Error:', e.message));"
```

R2 outage: file uploads fail but core accounting functionality works. Log errors, retry uploads. R2 outages are typically < 15 min.

4.5 Email Delivery Failure

```
# Check SendGrid activity
# SendGrid Dashboard → Activity → Filter by date → Look for bounces/blocks

# Test email sending manually
railway run node -e "
const sgMail = require('@sendgrid/mail');
sgMail.setApiKey(process.env.SENDGRID_API_KEY);
sgMail.send({ to: 'alem@alai.no', from: 'noreply@bilko.io', subject: 'Test', text: 'Test' })
  .then(() => console.log('Email sent'))
  .catch(e => console.error('Email error:', e.message));"
```

Email failure: invoices cannot be sent. Users can still create and download invoices. Not P0 but resolve within 2h.

---

5. Maintenance Operations

5.1 Routine Backup Verification (Monthly)

```
# Verify backup exists and restore to staging
# Railway Dashboard → PostgreSQL → Backups → Select latest

# Download backup
railway run pg_dump $DATABASE_URL -f monthly_verify_$(date +%Y%m).dump

# Restore to staging and verify counts
railway run psql $STAGING_DB_URL < monthly_verify_$(date +%Y%m).dump
railway run psql $STAGING_DB_URL -c "SELECT COUNT(*) FROM organizations;"
railway run psql $STAGING_DB_URL -c "SELECT COUNT(*) FROM invoices;"
```

5.2 Certificate Renewal

Certificates auto-renew via Vercel (Let's Encrypt) and Railway. Monitor expiry dates:

```
echo | openssl s_client -connect bilko.io:443 2>/dev/null | openssl x509 -noout -dates
echo | openssl s_client -connect api.bilko.io:443 2>/dev/null | openssl x509 -noout -dates
```

Alert if expiry < 30 days.

5.3 Secret Rotation (Annual / On Compromise)

```
# Generate new JWT secrets
openssl rand -base64 32  # JWT_SECRET
openssl rand -base64 32  # JWT_REFRESH_SECRET

# Update in Railway (this will invalidate ALL existing sessions)
railway variables set JWT_SECRET=<new> --service api --environment production
railway variables set JWT_REFRESH_SECRET=<new> --service api --environment production
railway service restart --service api

# Notify users: all sessions invalidated, need to log in again
```

---

6. Useful Commands Reference

```
# Railway CLI quick reference
railway login                          # Authenticate
railway status                         # Project status
railway logs --service api             # Stream logs
railway run <command>                  # Run in Railway context
railway variables list --service api   # Show env vars
railway service restart --service api  # Restart service

# Database quick reference
railway run psql $DATABASE_URL         # Connect to database
railway run npx prisma studio          # Database GUI (opens on localhost:5555)
railway run npx prisma migrate deploy  # Apply pending migrations

# Vercel quick reference
vercel ls                              # List deployments
vercel --prod                          # Deploy to production
vercel rollback                        # Rollback to previous
```

Related Documents

- Monitoring & Observability
- Disaster Recovery Plan
- Incident Report
- Go-Live Runbook

Approval

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

Post-Mortem Template

Post-Mortem

> **Project:** Bilko **Version:** 0.1 **Date:** 2026-02-23 **Author:** Ops Architect **Status:** Draft (Template — fill in per P0 incident) **Reviewers:** Tech Lead, Alem Bašić

Document History

Version	Date	Author	Changes
0.1	2026-02-23	Ops Architect	Initial draft

INSTRUCTIONS

Post-mortems are required for all P0 incidents and recommended for P1. Schedule within 5 business days of incident resolution.

Blameless culture: This document is about systems and processes, not people. The goal is to learn and prevent recurrence, not to assign blame.

Create a new file: `POST-MORTEM-YYYY-MM-DD-<title>.md` in `docs/operations/post-mortems/`

Post-Mortem: [INCIDENT TITLE]

Post-Mortem Date: YYYY-MM-DD **Incident Date:** YYYY-MM-DD **Incident Reference:** INC-YYYY-MM-DD-NNN **Facilitator:** [Name] **Attendees:** [Names] **Duration of post-mortem session:** [X minutes]

---

1. Executive Summary

What happened: [2-3 sentences: the incident, impact, and resolution]

Why it happened: [1-2 sentences: root cause in plain language]

What we're doing to prevent recurrence: [1-2 sentences: top action items]

---

2. Impact Summary

Metric	Value
Incident duration	X hours Y minutes
Detection time	X minutes from first symptom
Response time	X minutes from alert to first action
Users impacted	[All / Specific org / None]
Financial records affected	[None / Describe]
Downtime cost (est.)	[€X in lost productivity / TBD]
GDPR breach notification required	[Yes / No]

---

3. Timeline (Detailed)

Time (CET)	Event	Who	Notes
HH:MM	[Event]	[Person]	[Notes]

Key timestamps:

- **First symptom:** HH:MM
- **Alert fired:** HH:MM (detection lag: X min)
- **Incident declared:** HH:MM (response lag: X min)
- **Root cause identified:** HH:MM (diagnosis duration: X min)
- **Fix applied:** HH:MM
- **Service restored:** HH:MM
- **Incident closed:** HH:MM
- **Total user impact duration:** X min

---

4. Root Cause Analysis

What happened technically

[Detailed technical explanation of the failure chain. Be specific: which component, which code path, which query.]

For Bilko financial incidents, this section must include:

- Which accounting module was affected (VAT / double-entry / invoice calc / currency)
- Was any financial data corrupted? If yes, which organizations, which time window
- Were NUMERIC(19,4) values preserved correctly during the incident?

Why it happened

[The "5 Whys" — trace back to the systemic cause]

1. **Why** did users lose access? → API returned 503 errors
2. **Why** did the API return 503? → Railway service restarted due to OOM
3. **Why** did the service run out of memory? → Invoice PDF generation loaded entire result set into memory
4. **Why** did we not catch this? → No memory profiling in development, load testing not done
5. **Why** was there no load testing? → No performance test plan existed

Root cause (systemic): [e.g., "Lack of memory usage monitoring and load testing prior to feature launch"]

Contributing Factors

Factor	Category	Severity
[Factor]	[Process / Code / Infrastructure / Communication]	[High / Med / Low]

---

5. What Went Well

- [e.g., BetterStack alert fired within 2 minutes of downtime starting]
- [e.g., Rollback procedure was documented and worked on first try]
- [e.g., Financial data integrity was preserved — no accounting records corrupted]
- [e.g., User communication was clear and timely]

---

6. What Went Poorly

- [e.g., No memory usage alerting was configured before launch]
- [e.g., The runbook did not cover OOM scenarios]
- [e.g., Detection took 8 minutes because uptime check interval was 5 min]
- [e.g., Only one person knew how to access Railway logs]

---

7. Action Items

High Priority (P0/P1 — complete within 2 weeks)

#	Action	Category	Owner	Due	Status
1	[Action]	Prevention	[Name]	YYYY-MM-DD	Open
2	[Action]	Detection	[Name]	YYYY-MM-DD	Open

Medium Priority (P2 — complete within 1 month)

#	Action	Category	Owner	Due	Status
3	[Action]	Process	[Name]	YYYY-MM-DD	Open

Low Priority (P3 — add to backlog)

#	Action	Category	Owner	Due	Status
4	[Action]	Nice-to-have	[Name]	Backlog	Open

Action categories: Prevention, Detection, Response, Documentation, Process, Tooling

---

8. Lessons Learned

Technical Lessons

[What did we learn about the technology, the system design, or the code?]

For Bilko financial system incidents:

- [e.g., NUMERIC(19,4) Decimal arithmetic must be tested under concurrent load]
- [e.g., VAT calculation should be validated server-side even if client sends computed totals]

Process Lessons

[What did we learn about our operations process, monitoring, or communication?]

Culture Lessons

[What did we learn about team practices, communication patterns, or organizational factors?]

---

9. Metrics Targets for Next Quarter

Based on this incident, these metrics are now tracked:

Metric	Current	Target	By
Mean time to detect (MTTD)	X min	< 3 min	YYYY-MM-DD
Mean time to respond (MTTR)	X min	< 10 min	YYYY-MM-DD
Mean time to resolve (MTTR)	X min	< 60 min	YYYY-MM-DD

---

10. Follow-Up Schedule

- ☐ Action items tracked in GitHub Issues (label: `post-mortem-action`)
- ☐ 2-week check-in: verify P0/P1 actions completed
- ☐ 1-month check-in: verify P2 actions completed
- ☐ Next post-mortem: review if similar incidents recurred

Approval

Role	Name	Date	Signature
Facilitator			
Reviewer	Alem Bašić		

SLA Report

SLA Report

> **Project:** Bilko **Version:** 0.1 **Date:** 2026-02-23 **Author:** Ops Architect **Status:** Draft (Template — fill in monthly) **Reviewers:** Tech Lead, Alem Bašić

Document History

Version	Date	Author	Changes
0.1	2026-02-23	Ops Architect	Initial draft

INSTRUCTIONS

Generate monthly SLA reports by the 5th business day of the following month. File location: `docs/operations/sla-reports/SLA-YYYY-MM.md`

SLA Report: [Month YYYY]

Reporting Period: YYYY-MM-01 to YYYY-MM-[last day] **Report Date:** YYYY-MM-DD **Prepared by:** Ops Architect

---

1. SLA Summary

Service Level Objectives (SLOs)

SLO	Target	Actual	Status

API availability	≥ 99.5% / month	X.XX%	□ / □
API P95 response time	< 500ms	XXXms	□ / □
API error rate (5xx)	< 0.5%	X.XX%	□ / □
Frontend availability	≥ 99.9% / month	X.XX%	□ / □
Uptime (combined)	≥ 99.5% / month	X.XX%	□ / □

SLO Calculation

```
API Availability = (Total minutes in month - Downtime minutes) / Total minutes × 100

Total minutes in month (28 days) = 40,320
Total minutes in month (31 days) = 44,640

Allowed downtime at 99.5%:
- 28-day month: 201.6 minutes = ~3h 22min
- 31-day month: 223.2 minutes = ~3h 43min
```

---

2. Uptime Metrics

API (api.bilko.io)

Metric	Value
Measured uptime	X.XX%
Total downtime	X minutes
Number of incidents	X
Longest outage	X minutes

Frontend (bilko.io)

Metric	Value
Measured uptime	X.XX%
Total downtime	X minutes
Number of incidents	X

Source: BetterStack uptime monitoring (1-min check interval)

---

3. Performance Metrics

API Response Times

Metric	Target	Week 1	Week 2	Week 3	Week 4	Month Avg
P50	< 100ms					
P95	< 500ms					
P99	< 1000ms					

Critical Endpoint Performance

Endpoint	P95 Target	P95 Actual	Status
POST /api/v1/invoices	< 500ms	XXXms	□ / □
GET /api/v1/invoices	< 200ms	XXXms	□ / □
GET /api/v1/reports/vat	< 3000ms	XXXms	□ / □
POST /api/v1/auth/login	< 300ms	XXXms	□ / □

Source: Railway metrics + Sentry performance monitoring

---

4. Error Metrics

Error Rate by Week

Week	Total Requests	5xx Errors	Error Rate	Status
Week 1			X.XX%	
Week 2			X.XX%	
Week 3			X.XX%	
Week 4			X.XX%	
Month			**X.XX%**	

Top Errors (Sentry)

#	Error	Count	Affected Users	Status
1	[Error message]	X	X	Fixed / Investigating
2				
3				

---

5. Incidents This Month

Incident ID	Date	Duration	Severity	Root Cause	Resolved
INC-YYYY-MM-DD-001	YYYY-MM-DD	X min	P0/P1/P2	[Short description]	Yes

Total downtime from incidents: X minutes **P0 incidents:** X (target: 0) **P1 incidents:** X (target: < 2/month)

---

6. Financial Data Integrity (Monthly Verification)

Required: Verify no financial data corruption occurred this month.

Check	Method	Result
Double-entry balance	SQL: SUM(debits) = SUM(credits) per org	□ Balanced / □ Issues found
Invoice total accuracy	SQL: total = subtotal + tax - discount	□ Accurate / □ Issues found
VAT calculation accuracy	Spot-check 10 random invoices	□ Accurate / □ Issues found
No orphaned transactions	SQL: all transactions have debit+credit	□ Clean / □ Issues found

Verification queries run on: YYYY-MM-DD **Verified by:** [Name]

```
-- Monthly double-entry balance verification
SELECT
  o.name as organization_name,
  SUM(CASE WHEN te.type = 'debit' THEN te.amount ELSE 0 END) as total_debits,
```

```
  SUM(CASE WHEN te.type = 'credit' THEN te.amount ELSE 0 END) as total_credits,
  ABS(SUM(CASE WHEN te.type = 'debit' THEN te.amount ELSE -te.amount END)) as imbalance
FROM transaction_entries te
JOIN transactions t ON t.id = te."transactionId"
JOIN organizations o ON o.id = t."organizationId"
WHERE t.created_at >= DATE_TRUNC('month', CURRENT_DATE)
  AND t.created_at < DATE_TRUNC('month', CURRENT_DATE) + INTERVAL '1 month'
GROUP BY o.id, o.name
HAVING ABS(SUM(CASE WHEN te.type = 'debit' THEN te.amount ELSE -te.amount END)) > 0.0001
ORDER BY imbalance DESC;
-- Expected: 0 rows (all organizations balanced)
```

---

7. Infrastructure Metrics

Railway (Backend + Database)

Resource	Average	Peak	Trend
API CPU	X%	X%	Stable / Growing / Decreasing
API Memory	XMB	XMB	Stable / Growing / Decreasing
DB Connections	X avg	X peak	Stable / Growing
DB Storage	XGB	—	+X GB this month

Vercel (Frontend)

Metric	Value
Total page views	X
Unique visitors	X
Average LCP	Xms
Average CLS	X

---

8. Cost Report

Service	Budget	Actual	Variance
Railway (API + DB)	€20	€XX	+/-€XX
Vercel	€0	€XX	+/-€XX
Cloudflare R2	€1	€XX	+/-€XX
SendGrid	€0	€XX	+/-€XX
Total	**€21**	**€XX**	**+/-€XX**

---

9. SLA Trending

Month	API Uptime	P95 Latency	Error Rate	Incidents
[Previous -2]	—	—	—	—
[Previous -1]	—	—	—	—
[This month]	X.XX%	XXXms	X.XX%	X

---

10. Action Items from This Report

#	Issue	Action	Owner	Due
1	[Issue]	[Action]	[Owner]	YYYY-MM-DD

Approval

Role	Name	Date	Signature
Author	Ops Architect		
Reviewer	Alem Bašić		