Test Inventory

Bilko — Test Inventory

Status: Partially stale — recount required after Kotlin/Ktor migration and Playwright partitioning
Version: 2.1 **Last Updated:** 2026-05-21 **Author:** ALAI Documentation Team

This inventory catalogs implemented tests in Bilko, organized by package and file. It currently contains historical Express/Prisma-era details and must not be used as the source of truth for current test counts. For current policy, use TEST-STRATEGY.md, E2E-TEST-PLAN.md, and DEMO-TESTING-PLAN.md.

Summary

> **Refresh note (2026-05-21):** quick filesystem inventory found `apps/api/src/test/kotlin` with many Kotlin test files, `packages/core/tests`, and `apps/e2e/tests`. The table below is historical until regenerated from the current tree.

> **Round 12.3 audit (2026-08-08, MC #100289):** filesystem cross-check confirms the table below is stale. `apps/api-express` (and its `tests/`, `tests/unit/`, `tests/e2e/`, `tests/integration/` vitest suites) is gone — deleted 2026-05-02 per MC #10493 / ADR-020+ADR-021 CEO directive. The canonical backend is Kotlin/Ktor at `apps/api/src/test/kotlin` (288 files, 4062 `@Test` cases — many are direct ports of the old Express suite, e.g. `InvoiceRoutesTest.kt` header cites "Port of: apps/api-express/tests/invoices.test.ts"). Two suites present on disk are **not listed in this table at all**: `apps/e2e/tests` (41 Playwright specs) and `apps/web` (43 frontend test files). Full gap table: Audit — Round 12.3.

Package	Test Files	Test Cases	Status

~~`apps/api/tests/` (mock suite)~~	~~11~~ → **0**	~~~130~~	**REMOVED** — api-express deleted (MC #10493)
~~`apps/api/tests/unit/` (unit suite)~~	~~4~~ → **0**	~~~60~~	**REMOVED** — api-express deleted
~~`apps/api/tests/e2e/` (E2E suite)~~	~~2~~ → **0**	~~~30~~	**REMOVED** — api-express deleted
~~`apps/api/tests/integration/` (real DB)~~	~~5~~ → **0**	~~~50~~	**REMOVED** — api-express deleted
`packages/core/tests/` (unit)	5 test files	121	**Verified accurate** (20+32+22+24+23=121)
`apps/api/src/test/kotlin` (Kotlin/Ktor, canonical)	288 test files	4062	**Implemented — was undocumented**
`apps/e2e/tests/` (Playwright)	41 test files	not counted this round	**Implemented — was undocumented**
`apps/web/` (frontend unit/component)	43 test files	not counted this round	**Implemented — was undocumented**
Total (current, verified 2026-08-08)	**377**	**≥4183**	Active

Test Category Breakdown

Category	Description	Requires DB
Mock suite (`tests/*.test.ts`)	API endpoint tests with mocked Prisma — fast, no DB	No
Unit suite (`tests/unit/`)	Service-layer tests, financial logic, SEF client	No
E2E suite (`tests/e2e/`)	End-to-end billing flows with mocked services	No
Integration suite (`tests/integration/`)	Real DB tests (docker-compose.test.yml)	**Yes** — PostgreSQL
Core unit (`packages/core/`)	Pure business logic — no HTTP, no DB	No

Audit — Round 12.3, documented vs implemented (2026-08-08)

MC #100289. Method: filesystem cross-reference (`find`/`grep -c`) against every row below this doc claimed, run against the current `azdo/main` checkout at `~/business/ALAI-Holding-AS/products/Bilko`. No coverage tool was run — "coverage %" targets in the table further below are unverified by this

round.

Documented claim (this file)	Verification method	Result
`apps/api/tests/`, `tests/unit/`, `tests/e2e/`, `tests/integration/` — 22 files, ~270 cases (Express/Prisma/vitest)	`ls apps/api/tests`, `ls apps/api-express`	**GAP — directory does not exist.** `apps/api-express` deleted 2026-05-02 (MC #10493). Only a stray `node_modules/` remains, no source or tests.
`packages/core/tests/` — 5 files, 121 cases (`accounting`=20, `chart-of-accounts`=32, `invoicing`=22, `multi-currency`=24, `tax`=23)	`grep -cE '^\s*(it|test)\('` per file	**MATCH.** All 5 files present, all 5 per-file counts exact, sum = 121 exact.
`invoicing.test.ts` "generateInvoiceNumber" — 8 tests	Read `describe('generateInvoiceNumber', …)` block	**MATCH.** 8 `it()` blocks (format, increment, pad, >999, empty prefix, whitespace prefix, negative lastNumber, non-integer lastNumber).
Coverage Tracking table: "Multi-tenant isolation (real DB) — Tests implemented (5 scenarios) — target 100%"	grep for the cited file `tests/integration/tenant-isolation.integration.test.ts`	**GAP — cited file does not exist** (Express deleted). Capability is NOT absent though: Kotlin suite has ≥6 dedicated files — `OrgScopeIsolationTest.kt`, `CrossTenantIsolationTest.kt`, `CrossTenantMt03P1Test.kt`, `RlsOrgIsolationV46IntegrationTest.kt`, `BankingServiceOrgIsolationTest.kt`, `AccountantPortalCrossOrgTest.kt`. Actual line/branch coverage % not measured this round — needs a `kover`/gradle coverage run to confirm the "100%" target, not claimed here.
Everything under `apps/api/src/test/kotlin` (canonical backend)	`find … -iname '*.kt'` + `grep -rc '@Test'`	**UNDOCUMENTED.** 288 `.kt` test files, 285 contain `@Test` (3 are shared base/helper classes: `DbTestBase.kt`, `TestModuleHelper.kt`, `StubCountryPlugin.kt`), 4062 `@Test` annotations total. None of this appears anywhere in this file before this round.
`apps/e2e/tests/` (Playwright E2E)	`find apps/e2e/tests -iname '*.spec.ts'`	**UNDOCUMENTED.** 41 spec files on disk, zero mentioned in this inventory. Covers auth lifecycle, invoice wizard, RBAC boundaries, multi-tenant isolation, SEF submission, storno flow, etc.
`apps/web/` (frontend unit/component tests)	`find apps/web -iname '*.test.ts(x)' -o -iname '*.spec.ts(x)'`	**UNDOCUMENTED.** 43 test files on disk, zero mentioned in this inventory.

Net verdict: the only section of this document that is still structurally and numerically correct is `packages/core/tests/`. Every `apps/api/*` row describes a backend that was deleted three months ago; the replacement backend (Kotlin/Ktor, 288 files / 4062 test cases) plus two whole suites (

`apps/e2e`, `apps/web`, 84 files combined) exist on disk but were never added to this document. This matches the file's own top-of-doc disclaimer ("Partially stale... must not be used as the source of truth") — this audit is the first line-by-line confirmation of that disclaimer with real numbers.

Not done this round (would require running test suites, out of scope for a doc audit): per-file Kotlin test-case breakdown beyond the aggregate `@Test` count, per-file `apps/e2e` and `apps/web` case counts, actual coverage percentages, and whether the `@Tag("integration")` Kotlin tests referenced in file headers (e.g. `InvoiceRoutesTest.kt`) run in CI.

`packages/core/tests/` — Unit Tests

Pure unit tests for the `@bilko/core` financial engine. No database, no HTTP. Uses `globals: true` (no explicit imports of `describe`/`it`/`expect`).

`accounting.test.ts` — 20 tests

Tests double-entry bookkeeping engine: `validateDoubleEntry`, `createJournalEntry`, `calculateTrialBalance`.

Test Name	What It Tests	Status
`validateDoubleEntry - returns true for balanced entry`	Debit total equals credit total	Implemented
`validateDoubleEntry - returns false for unbalanced entry`	Debit ≠ credit	Implemented
`validateDoubleEntry - returns false for fewer than 2 lines`	Minimum 2 lines required	Implemented
`validateDoubleEntry - returns false for empty lines`	Empty array → false	Implemented
`validateDoubleEntry - returns false for negative amounts`	Negative amounts invalid	Implemented
`validateDoubleEntry - returns false for zero amounts`	Zero amounts invalid	Implemented
`validateDoubleEntry - handles multiple lines that sum to balanced`	Multi-line balanced entry	Implemented
`validateDoubleEntry - handles decimal amounts with precision`	NUMERIC(19,4) precision	Implemented
`createJournalEntry - returns entry when valid and balanced`	Happy path	Implemented
`createJournalEntry - throws for fewer than 2 lines`	Error: "at least 2 lines"	Implemented
`createJournalEntry - throws for empty lines array`	Error: "at least 2 lines"	Implemented

Test Name	What It Tests	Status
`createJournalEntry - throws for missing description`	Error: "must have a description"	Implemented
`createJournalEntry - throws for whitespace-only description`	Whitespace = missing	Implemented
`createJournalEntry - throws for missing date`	Error: "must have a date"	Implemented
`createJournalEntry - throws for unbalanced entry`	Error shows debit vs credit amounts	Implemented
`createJournalEntry - throws for negative amount lines`	Negative amounts rejected	Implemented
`calculateTrialBalance - returns balanced from balanced entries`	isBalanced=true, totals correct	Implemented
`calculateTrialBalance - groups by account number`	Rows aggregated per account	Implemented
`calculateTrialBalance - returns empty rows for no transactions`	Empty array → zero totals	Implemented
`calculateTrialBalance - sorts rows by account number`	Rows sorted ascending	Implemented

`chart-of-accounts.test.ts` — 32 tests

Tests chart of accounts operations: account creation, parent-child hierarchy, account type validation.

Test Area	Test Count	Status
Account creation (valid + invalid inputs)	~10	Implemented
Account hierarchy (parent-child relationships)	~8	Implemented
Account type validation (Asset/Liability/Equity/Revenue/Expense)	~7	Implemented
Account code format validation (1xxx-5xxx range)	~7	Implemented

---

`invoicing.test.ts` — 22 tests

Tests invoice number generation, total calculations, and line item validation.

Test Name	What It Tests	Status

`generateInvoiceNumber - generates INV-YYYY-NNN format`	Standard format	Implemented
`generateInvoiceNumber - increments from last number`	Sequential numbering	Implemented
`generateInvoiceNumber - pads number to 3 digits`	Zero-padding	Implemented
`generateInvoiceNumber - handles numbers beyond 999`	No truncation for 1000+	Implemented
`generateInvoiceNumber - throws for empty prefix`	Error: "prefix is required"	Implemented
`generateInvoiceNumber - throws for whitespace-only prefix`	Whitespace = invalid	Implemented
`generateInvoiceNumber - throws for negative lastNumber`	Error: "non-negative integer"	Implemented
`generateInvoiceNumber - throws for non-integer lastNumber`	Float rejected	Implemented
`calculateInvoiceTotals - calculates line item total`	quantity × unitPrice	Implemented
`calculateInvoiceTotals - calculates subtotal as sum of lines`	Sum of all line totals	Implemented
`calculateInvoiceTotals - calculates tax per line item`	lineTotal × taxRate / 100	Implemented
`calculateInvoiceTotals - calculates total = subtotal + tax`	Final total	Implemented
`calculateInvoiceTotals - handles items without taxRate`	Missing tax = 0	Implemented
`calculateInvoiceTotals - returns zeros for empty items`	Empty array → all zeros	Implemented
`calculateInvoiceTotals - handles multiple items with different rates`	Mixed 20%/10% tax	Implemented
`calculateInvoiceTotals - maintains decimal precision`	3 × 33.33 = 99.99	Implemented
`validateLineItem - returns true for valid item`	Happy path	Implemented
`validateLineItem - returns false for zero quantity`	qty=0 invalid	Implemented
`validateLineItem - returns false for negative quantity`	qty<0 invalid	Implemented
`validateLineItem - returns false for negative unitPrice`	price<0 invalid	Implemented
`validateLineItem - returns false for empty description`	Description required	Implemented
`validateLineItem - returns false for whitespace-only description`	Whitespace = empty	Implemented

`multi-currency.test.ts` — 24 tests

Tests currency conversion, exchange rate locking, and NUMERIC precision handling.

Test Area	Test Count	Status
Currency conversion (EUR/RSD/BAM)	~8	Implemented
Exchange rate locking at transaction date	~6	Implemented
NUMERIC(19,4) precision (no float drift)	~5	Implemented
Edge cases (zero rate, missing rate, same currency)	~5	Implemented

---

`tax.test.ts` — 23 tests

Tests VAT calculations for all supported countries and edge cases.

Test Area	Test Count	Status
Serbia PDV (20% standard, 10% reduced, 0% exempt)	~6	Implemented
BiH PDV (17% standard)	~4	Implemented
Croatia PDV (25% standard, 13% reduced, 5% super-reduced)	~5	Implemented
Mixed tax rates on single invoice	~3	Implemented
Zero-rate exports	~2	Implemented
Reverse VAT / gross-to-net	~3	Implemented

`apps/api/tests/` — Mock API Tests

Integration tests for Express API endpoints. Tests use **mocked Prisma client** — no real database required. Setup in `tests/setup.ts`.

`setup.ts` — Test Infrastructure (not a test file)

Provides:

- Prisma client mock via `vi.mock('../src/lib/prisma')`
- Environment variable setup (JWT secrets, rate limits)
- `createTestUser()` — factory for test user objects
- `generateTestAccessToken()` — valid JWT for authenticated requests
- `generateTestRefreshToken()` — valid refresh token
- Constants: `TEST_ORG_ID`, `TEST_USER_ID`, `TEST_USER_EMAIL`

`auth.test.ts` — 11 tests

Test Name	Route	Status
`returns 201 with user, organization, and tokens`	POST /auth/register	Implemented
`returns 400 for duplicate email`	POST /auth/register	Implemented
`returns 200 with tokens for valid credentials`	POST /auth/login	Implemented
`returns 401 for invalid password`	POST /auth/login	Implemented
`returns 401 for non-existent email`	POST /auth/login	Implemented
`returns 200 with new access token for valid refresh token`	POST /auth/refresh	Implemented
`returns 401 when no refresh token cookie is present`	POST /auth/refresh	Implemented
`returns 204 and clears cookie`	POST /auth/logout	Implemented
`returns 200 with current user when authenticated`	GET /auth/me	Implemented
`returns 401 when no token provided`	GET /auth/me	Implemented
`returns 401 for invalid token`	GET /auth/me	Implemented

`invoices.test.ts` — 11 tests

Test Name	Route	Status
`returns 200 with paginated list`	GET /invoices	Implemented
`returns 401 without auth`	GET /invoices	Implemented
`returns 201 with created invoice`	POST /invoices	Implemented
`returns 200 with full invoice`	GET /invoices/:id	Implemented
`returns 404 when invoice not found`	GET /invoices/:id	Implemented
`returns 200 when updating draft invoice`	PUT /invoices/:id	Implemented

Test Name	Route	Status
`returns 400 when updating sent invoice`	PUT /invoices/:id	Implemented
`returns 200 when sending invoice (draft -> sent)`	PATCH /invoices/:id/status	Implemented
`returns 200 when marking invoice as paid (sent -> paid)`	PATCH /invoices/:id/status	Implemented
`returns 204 when deleting draft invoice`	DELETE /invoices/:id	Implemented
`returns 400 when deleting sent invoice`	DELETE /invoices/:id	Implemented

`expenses.test.ts` — 9 tests

Test Area	Route	Status
List expenses (200 + auth)	GET /expenses	Implemented
Create expense (201)	POST /expenses	Implemented
Get expense by ID (200 + 404)	GET /expenses/:id	Implemented
Update expense (200 + 400 for non-pending)	PUT /expenses/:id	Implemented
Approve expense (200 + role check)	PATCH /expenses/:id/approve	Implemented
Delete expense (204 + 400 for approved)	DELETE /expenses/:id	Implemented

---

`contacts.test.ts` — 9 tests

Test Area	Route	Status
List contacts (200 + auth)	GET /contacts	Implemented
Create contact (201 + validation)	POST /contacts	Implemented
Get contact (200 + 404)	GET /contacts/:id	Implemented
Update contact (200)	PUT /contacts/:id	Implemented
Delete contact (204)	DELETE /contacts/:id	Implemented

`accounts.test.ts` — 4 tests

Test Area	Route	Status
List chart of accounts (200)	GET /accounts	Implemented
Get account by ID (200 + 404)	GET /accounts/:id	Implemented
Create account (201)	POST /accounts	Implemented
Delete account (204)	DELETE /accounts/:id	Implemented

`banking.test.ts` — 10 tests

Test Area	Route	Status
List bank accounts (200)	GET /banking/accounts	Implemented
Create bank account (201)	POST /banking/accounts	Implemented
Import bank statement (200 + validation)	POST /banking/accounts/:id/import	Implemented
List bank transactions (200)	GET /banking/accounts/:id/transactions	Implemented
Reconcile transaction (200 + 400 for mismatch)	POST /banking/accounts/:id/reconcile	Implemented

---

`reports.test.ts` — 9 tests

Test Area	Route	Status
Profit & Loss report (200 + date range)	GET /reports/profit-loss	Implemented
Balance sheet (200)	GET /reports/balance-sheet	Implemented
VAT report for RS (200 + correct rates)	GET /reports/vat	Implemented
Trial balance (200)	GET /reports/trial-balance	Implemented
Auth required on all report endpoints	All report routes	Implemented

---

`transactions.test.ts` — 9 tests

Test Area	Route	Status
List transactions (200 + pagination)	GET /transactions	Implemented
Filter by account (200)	GET /transactions?accountId=	Implemented

Test Area	Route	Status
Get transaction by ID (200 + 404)	GET /transactions/:id	Implemented
Auth required	All transaction routes	Implemented

`country.test.ts` — 27 tests

Tests the country plugin integration — routes that return country-specific tax configuration.

Test Area	Route	Status
Serbian PDV rates (20/10/0) for RS org	GET /country/tax-rates	Implemented
Bosnian PDV rate (17) for BA org	GET /country/tax-rates	Implemented
Croatian PDV rates (25/13/5/0) for HR org	GET /country/tax-rates	Implemented
Invoice number format per country	GET /country/invoice-format	Implemented
Auth required	All country routes	Implemented
Unknown country code (400)	GET /country/tax-rates	Implemented

---

`chatbot.test.ts` — Chatbot API Tests

Test Name	Route	Status
`returns 200 with assistant response`	POST /chatbot/message	Implemented
`returns 400 when message is empty`	POST /chatbot/message	Implemented
`returns 429 when rate limit exceeded`	POST /chatbot/message	Implemented
`returns 401 without auth`	POST /chatbot/message	Implemented
`returns 200 with conversation history`	GET /chatbot/history	Implemented
`returns empty array when no history exists`	GET /chatbot/history	Implemented
`returns 401 without auth on history`	GET /chatbot/history	Implemented
`returns 204 when history cleared`	DELETE /chatbot/history	Implemented
`returns 401 without auth on clear history`	DELETE /chatbot/history	Implemented

`invoice-gl-reversal.test.ts` — Invoice GL Reversal Tests

Tests `InvoiceService.cancelInvoice()` — when a SENT invoice is cancelled, reversing double-entry GL entries are created to undo the original booking.

Test Area	Status
Cancels draft invoice (sets status to cancelled, no GL reversal)	Implemented
Cancels sent invoice (creates reversing GL transactions)	Implemented
Reversal debits = original credits (GL stays balanced)	Implemented
Throws 404 when invoice not found	Implemented
Throws 400 when invoice is already cancelled	Implemented
Throws 400 when invoice is paid (cannot cancel paid invoices)	Implemented

---

`new-endpoints.test.ts` — Additional Endpoint Tests

Tests for supplemental endpoints not covered in the main mock suite.

Test Area	Route	Status
Receipt not attached (404)	GET /expenses/:id/receipt	Implemented
Receipt auth required (401)	GET /expenses/:id/receipt	Implemented
VAT export PDF (200 / 404)	GET /reports/vat/export/pdf	Implemented
VAT export XML (200 / 404)	GET /reports/vat/export/xml	Implemented
Dashboard metrics (200)	GET /reports/dashboard	Implemented
Dashboard auth required (401)	GET /reports/dashboard	Implemented
Audit log (200 + owner/admin only)	GET /security/audit-log	Implemented
Audit log role check (403 for viewer)	GET /security/audit-log	Implemented
Data export (200 + owner only)	POST /security/data-export	Implemented
Data export role check (403 for admin)	POST /security/data-export	Implemented

---

`e2e/api.test.ts` — Full E2E (no mocks, live server)

End-to-end API integration test. Exercises the full Express application stack with a live server.

Status
Implemented

`e2e/billing-flow.e2e.test.ts` — Billing Workflow E2E

Tests the full billing flow through HTTP endpoints with mocked services (no real DB required):

1. Create contact
2. Create invoice
3. Send invoice (draft → sent)
4. Mark invoice paid (sent → paid)
5. Check P&L shows revenue
6. Verify trial balance returns `isBalanced=true`
7. Multi-currency invoice in EUR with country VAT rates
8. Credit note creation

Status
Implemented

`apps/api/tests/unit/` — Unit Tests (service layer)

Service-level unit tests with mocked Prisma. No HTTP layer. Tests business logic in individual service classes.

`invoice-service-calculations.test.ts` — Invoice Arithmetic

Tests `InvoiceService.createInvoice()` arithmetic at the service layer. Verifies:

- `lineTotal = quantity × unitPrice`
- `lineTax = lineTotal × taxRate / 100`
- `subtotal = sum(lineTotals)`
- `taxAmount = sum(lineTaxes)`
- `total = subtotal + taxAmount`
- `baseAmount = total × exchangeRate`

Test Area	Status
Single-line invoice arithmetic	Implemented
Multi-line invoice with mixed tax rates	Implemented
Multi-currency exchange rate application	Implemented
Zero-quantity line items rejected	Implemented
NUMERIC(19,4) precision maintained	Implemented

`two-factor.test.ts` — Two-Factor Authentication Service

Tests `TwoFactorService` at the service level. Mocks: bcryptjs, speakeasy, qrcode, Prisma.

Test Name	Status
`enable - wrong password → throws unauthorized`	Implemented
`enable - correct password → returns secret + QR data URL + 10 codes`	Implemented
`enable - backup codes are hashed before storing`	Implemented
`enable - plaintext backup codes returned once only`	Implemented
`verify - valid TOTP + window=1 → activates 2FA`	Implemented
`verify - invalid TOTP → throws badRequest`	Implemented
`disable - correct password → clears secret + backup codes`	Implemented
`disable - wrong password → throws unauthorized`	Implemented

`sef-submission.test.ts` — SEF (Serbia E-Invoicing) Client

Tests `SefClient` class and `InvoiceService.submitToSef()` fire-and-forget behavior. HTTP calls are mocked via `vi.spyOn(global, 'fetch')`.

Test Area	Status
`SefClient` submits UBL XML to SEF sandbox URL	Implemented
`SefClient` uses production URL in production env	Implemented
`SefClient` retries on network failure	Implemented
`createSefClientFromEnv()` reads credentials from env vars	Implemented
`generateSEFInvoiceXML()` produces valid UBL 2.1 XML	Implemented
`InvoiceService.submitToSef()` never re-throws errors	Implemented

`vat-calculation.test.ts` — VAT Calculation Tests (Country Packages)

Tests pure calculation functions from `@bilko/country-rs`, `@bilko/country-ba`, and `@bilko/country-hr`. No Prisma, no HTTP — stateless math functions.

Country	Functions Tested	Status
Serbia	`calculateSerbianPDV`, `calculateNetFromGrossPDV`, `qualifiesForPausalRegime`, `requiresVATRegistration`, `calculateSerbianCIT`	Implemented
Bosnia	`calculateBosnianPDV`, `calculateNetFromGrossPDV`, `requiresVATRegistration`	Implemented
Croatia	`calculateCroatianPDV`, `calculateNetFromGrossPDV`, `requiresVATRegistration`	Implemented
All rates	Standard, reduced, zero, super-reduced rates per country	Implemented

---

`apps/api/tests/integration/` — Real Database Tests

Integration tests that run against a **real PostgreSQL database** via `docker-compose.test.yml`. Requires Docker.

Run with:

```
cd apps/api
docker-compose -f ../../docker-compose.test.yml up -d
npm run test:integration
```

`auth.integration.test.ts` — Auth Integration

Full registration + login + refresh flow against real DB.

Test Area	Status
Register new organization + owner user	Implemented
Login with correct credentials	Implemented
Refresh access token via cookie	Implemented
Reject duplicate email registration	Implemented

`invoice.integration.test.ts` — Invoice CRUD Integration

Invoice lifecycle against real DB: create → read → update → status change.

Test Area	Status
Create invoice with line items	Implemented
Read invoice by ID	Implemented
Update draft invoice	Implemented
Send invoice (draft → sent)	Implemented
Mark paid (sent → paid)	Implemented

`credit-note-gl.integration.test.ts` — Credit Note GL Integration

Tests credit note creation against real DB — verifies reversing GL entries balance.

Test Area	Status
Credit note creates reversing journal entries	Implemented
Reversing entries balance (debits = credits)	Implemented

Test Area	Status
Original invoice marked as credited	Implemented

`report.integration.test.ts` — Reports Integration

Report generation against real DB with seeded transactions.

Test Area	Status
P&L report returns revenue/expense data	Implemented
VAT report returns correct tax amounts	Implemented
Trial balance returns `isBalanced=true`	Implemented

`tenant-isolation.integration.test.ts` — Multi-Tenant Security

Verifies multi-tenant isolation: Organization A cannot read or modify Organization B's data.

Test Area	Status
Org A cannot list Org B's invoices	Implemented
Org A cannot read Org B's invoice by ID	Implemented
Org A cannot update Org B's invoice	Implemented
Org A cannot delete Org B's contact	Implemented
Cross-tenant requests return 404 (not 403 — no data leakage)	Implemented

---

Coverage Tracking

Module	Current Status	Target
`@bilko/core` accounting engine	Tests implemented (20 cases)	>95%
`@bilko/core` invoicing	Tests implemented (22 cases)	>95%
`@bilko/core` tax/VAT	Tests implemented (23 cases)	>95%

Module	Current Status	Target
`@bilko/core` multi-currency	Tests implemented (24 cases)	>90%
`@bilko/core` chart of accounts	Tests implemented (32 cases)	>90%
Auth API	Tests implemented (11 cases)	>85%
Invoices API	Tests implemented (11 cases)	>80%
Expenses API	Tests implemented (9 cases)	>80%
Banking API	Tests implemented (10 cases)	>75%
Reports API	Tests implemented (9 cases)	>75%
Country/Tax-Rates API	Tests implemented (27 cases)	>80%
Chatbot API	Tests implemented (9 cases)	>80%
Security/Audit API	Tests implemented (via new-endpoints)	>80%
Invoice GL Reversal (service layer)	Tests implemented	>90%
Two-Factor Auth (service layer)	Tests implemented (8 cases)	>90%
SEF Client + Submission	Tests implemented	>85%
VAT Calculations (country packages)	Tests implemented	>95%
Multi-tenant isolation (real DB)	Tests implemented (5 scenarios)	100%
Invoice CRUD (real DB)	Tests implemented	>80%
Credit note GL (real DB)	Tests implemented	>90%

Test Execution Commands

```
# All tests (from project root — mock + unit + E2E suites)
npm run test

# Core unit tests only
cd packages/core && npx vitest run

# API mock suite only (no DB required)
cd apps/api-express && npx vitest run

# API unit suite only
cd apps/api-express && npx vitest run tests/unit/
```

```
# API E2E suite only (mocked services, no DB)
cd apps/api-express && npx vitest run tests/e2e/

# Real DB integration tests (requires docker-compose.test.yml)
docker-compose -f docker-compose.test.yml up -d
cd apps/api-express && npm run test:integration

# Watch mode (re-run on change)
cd packages/core && npx vitest
cd apps/api-express && npx vitest

# Specific file
cd apps/api-express && npx vitest run tests/auth.test.ts
cd apps/api-express && npx vitest run tests/unit/two-factor.test.ts

# With coverage
cd packages/core && npx vitest run --coverage

# Verbose output
npx vitest run --reporter=verbose
```

Related Documents

- Testing Guide: TESTING-GUIDE.md
- Backend Architecture: ../backend/BACKEND-ARCHITECTURE.md

Last Updated: 2026-08-08 (Round 12.3 audit, MC #100289) **Status:** Partially stale — `apps/api/*` rows below the audit section are historical/removed; see Audit — Round 12.3 for verified current counts **Total Tests (verified 2026-08-08):** 377 test files — `packages/core` 5 (121 cases, exact), `apps/api/src/test/kotlin` 288 (4062 `@Test`), `apps/e2e/tests` 41, `apps/web` 43. Per-file breakdown for the Kotlin/e2e/web suites not yet documented — recommend a follow-up MC to regenerate detailed per-file tables the way `packages/core/tests/` is documented below.

Revision #24
Created 2026-02-18 08:44:52 UTC by John
Updated 2026-08-19 10:33:27 UTC by John