Database Schema

Bilko Database Schema

> **Status:** IMPLEMENTED **Last verified:** 2026-05-20 **Canonical backend:** `apps/api` Kotlin/Ktor service **Database:** PostgreSQL on GCP Cloud SQL for deployed environments **Schema source of truth:** Flyway SQL migrations + Exposed table mappings

This document describes the database schema currently used by the Bilko Kotlin/Ktor API. It replaces the older ORM-era database notes and must be kept aligned with:

- Flyway migrations: `apps/api/src/main/resources/db/migration/`
- Exposed table mappings: `apps/api/src/main/kotlin/no/alai/bilko/models/Tables.kt`
- Route/service behaviour: `apps/api/src/main/kotlin/no/alai/bilko/routes/` and `apps/api/src/main/kotlin/no/alai/bilko/services/`
- Environment mapping: `infrastructure/gcp/ENV-MATRIX.md`

Do not treat generated diagrams, frontend type definitions, or archived deployment notes as database authority.

---

1. Architecture Overview

Bilko is a multi-tenant accounting SaaS. The active backend stores tenant data in PostgreSQL and scopes business tables by `organization_id` where appropriate.

Runtime stack:

- **API:** Kotlin/Ktor
- **SQL migration engine:** Flyway
- **Kotlin SQL mapping:** JetBrains Exposed
- **Database engine:** PostgreSQL
- **Deployed DB platform:** GCP Cloud SQL
- **Primary migration command path:** Cloud Build / backend Gradle Flyway tasks

The schema is forward-only. Applied migrations must not be edited after deployment. If a deployed environment has Flyway metadata drift, repair is handled as a controlled operations procedure with target identity checks, schema checks, transcript, and postflight validation.

Relationship overview

```
erDiagram
  organizations ||--o{ users : owns
  organizations ||--o{ accounts : owns
  organizations ||--o{ contacts : owns
  organizations ||--o{ invoices : owns
  organizations ||--o{ expenses : owns
  organizations ||--o{ transactions : owns
  organizations ||--o{ bank_accounts : owns
  organizations ||--o{ recurring_invoices : owns
  organizations ||--o{ adapter_config : configures
  organizations ||--o{ stripe_webhook_events : receives

  users ||--o{ refresh_tokens : has
  users ||--o{ invoices : creates
  users ||--o{ expenses : creates
  users ||--o{ transactions : creates
  users ||--o{ logged_actions : actor

  account_types ||--o{ accounts : classifies
  accounts ||--o{ accounts : parent
  accounts ||--o{ invoice_items : revenue_account
  accounts ||--o{ expenses : expense_account
  accounts ||--o{ bank_accounts : ledger_account
  accounts ||--o{ transactions : debit_account
  accounts ||--o{ transactions : credit_account

  contacts ||--o{ invoices : customer
  contacts ||--o{ expenses : vendor
  contacts ||--o{ recurring_invoices : template_customer

  invoices ||--o{ invoice_items : contains
  bank_accounts ||--o{ bank_transactions : imports
  transactions ||--o{ bank_transactions : reconciles
  currencies ||--o{ exchange_rates : base_currency
```

```
    currencies ||--o{ exchange_rates : target_currency
```

---

2. Source-of-Truth Rules

1. **Add schema changes via new Flyway migrations only.**
   - Use the next available version under `apps/api/src/main/resources/db/migration/`.
   - Never rewrite an already-applied migration in demo, staging, or production.
2. **Update Exposed mappings in the same change.**
   - `Tables.kt` should match the migrated database surface used by routes/services.
3. **Update API and docs together.**
   - If a schema change alters request/response shapes, update `docs/backend/openapi.yaml` and relevant backend docs.
4. **Validate with Flyway before deploy promotion.**
   - A valid deploy target must pass Flyway validation before the API is considered healthy.
5. **Keep environment identity explicit.**
   - Staging, demo, and production databases are separate targets. Migration or repair work must state which database is being touched.

---

3. Migration Inventory

As of this verification pass, the repository contains **36 Flyway migration files**. The deployed stage repair for MC #101509 validated the current Flyway version as **35** after applying pending migrations.

Important migration groups:

Version range	Purpose
V1-V6	Initial accounting schema, compatibility columns, encrypted identifiers, supplementary tables, organization logo, role storage normalization
V7-V15	Plan tiers, Stripe webhook logging, compliance calendar, recurring invoices, audit enum cleanup, demo/CI seed data, trial fields
V16-V24	Country constraints, RLS permissive baseline, invoice/expense status enum support, demo org fixes, CI viewer, BA jurisdiction split, BA entity charts
V25-V29	Adapter configuration, platform-admin marker, Serbia chart, logged action width, UAT demo users

Version range	Purpose
V30-V35	RLS/session auth hardening, security-definer auth helpers, demo admin grants, bcrypt-prefix normalization, UAT password hash reset

Operational note from MC #101509:

- Staging checksum drift was detected for V22, V25, V26, and V28.
- Schema checks proved checksum-only drift before repair.
- Controlled Flyway repair + migrate brought staging to version 35 and `flyway validate` passed.
- The authoritative stage Cloud Build trigger then succeeded.

---

4. Tenant and Security Model

Most business data is scoped by `organization_id` and accessed through authenticated Ktor routes. The schema includes:

- Organization-level tenant boundary.
- User roles per organization.
- Platform-admin marker for controlled platform operations.
- Refresh-token session storage.
- Logged action/audit table.
- Row-level-security related migration work in the V17 and V30+ series.

Application code must set the correct organization/user context before querying tenant-scoped tables. Any new table containing tenant data should include `organization_id` unless it is intentionally global reference data.

---

5. Tables

The following table inventory is derived from `Tables.kt` and active migrations.

Column notation

The detailed type/constraint authority remains Flyway SQL plus `Tables.kt`. This document uses these shorthand types for quick review:

Notation	Meaning

`uuid pk`	UUID primary key
`uuid fk`	UUID foreign key
`text` / `varchar(n)`	String column; exact length is migration-defined
`numeric`	Decimal money/rate value
`date` / `timestamp`	Date/time column
`json/jsonb`	Structured JSON column
`bool`	Boolean
`soft-delete`	Nullable `deleted_at` lifecycle column

Common lifecycle columns are `created_at`, `updated_at`, `version`, and `deleted_at` where present.

5.1 `organizations`

Tenant root table.

Key fields:

- `id`
- `name`
- `registration_number`
- `vat_number`, `vat_country`, `vat_registered`, `vat_rate`
- `firm_type`
- `base_currency`
- `country`, `language`
- `fiscal_year_start`
- `logo_url`
- `security_settings`
- subscription/trial fields: `plan_tier`, `quota_invoices_month`, `quota_contacts`, `quota_users`, `stripe_customer_id`, `stripe_subscription_id`, `trial_started_at`, `trial_ends_at`
- lifecycle fields: `created_at`, `updated_at`, `version`, `deleted_at`

Column summary:

Column group	Type/constraint summary
Identity	`id uuid pk`, `name text`
Registration/tax	registration and VAT columns are nullable text values; VAT registration is boolean-backed
Locale/market	`country`, `language`, base currency, fiscal year start
Branding/settings	`logo_url`, `security_settings json/jsonb`
Subscription/trial	plan, quota, Stripe IDs, trial timestamps

Column group	Type/constraint summary
Lifecycle	timestamps, `version`, `deleted_at soft-delete`

Notes:

- `country` is constrained by migration history and used by market-specific tax/e-invoice logic.
- BA jurisdiction support was added in the V22/V23/V24 migration set.

5.2 `users`

Authenticated users belonging to organizations.

Key fields:

- `id`
- `organization_id`
- `email`
- `password_hash`
- `full_name`
- `role`
- `two_factor_enabled`, `two_factor_secret`, `two_factor_backup_codes`
- `notification_preferences`
- `last_login_at`
- invite fields: `invite_token`, `invite_expires_at`
- `status`
- `is_platform_admin`
- lifecycle fields: `created_at`, `updated_at`, `version`, `deleted_at`

Column summary:

Column group	Type/constraint summary
Identity	`id uuid pk`, `organization_id uuid fk`
Login	`email text unique`, `password_hash text`
RBAC	`role text`, `status text`, `is_platform_admin bool`
2FA	boolean flag, secret, backup-code storage
Invites/session metadata	invite token/expiry, last login timestamp
Lifecycle	timestamps, `version`, `deleted_at soft-delete`

Notes:

- Password and 2FA behaviour is implemented in the Kotlin auth services.

- `is_platform_admin` was added by V26.

5.3 `refresh_tokens`

Session refresh-token storage.

Key fields:

- `id`
- `user_id`
- `jti`
- `expires_at`
- `created_at`
- `version`

Notes:

- Used by auth-lifecycle and logout/revocation flows.
- Session listing/revocation routes are documented in OpenAPI.

5.4 `account_types`

Reference data for account classifications.

Key fields:

- `id`
- `name`
- `normal_balance`
- `created_at`
- `version`

5.5 `accounts`

Chart of accounts entries.

Key fields:

- `id`
- `organization_id`
- `code`
- `name`
- `account_type_id`
- `currency_code`

- `parent_account_id`
- `is_active`
- lifecycle fields: `created_at`, `updated_at`, `version`, `deleted_at`

Notes:

- Market-specific chart additions exist for BA and RS.
- Account hierarchy is represented with `parent_account_id`.

5.6 `contacts`

Customers and vendors.

Key fields:

- `id`
- `organization_id`
- `type`
- `name`
- `email`, `phone`
- registration and tax identifiers: `registration_number`, `vat_number`, `jmbg`, `jmbg_hash`, `oib`, `oib_hash`
- address fields: `address_line1`, `address_line2`, `city`, `postal_code`, `country`
- `currency_code`
- `payment_terms`
- `notes`
- `is_active`
- lifecycle fields: `created_at`, `updated_at`, `version`, `deleted_at`

Notes:

- Sensitive personal/business identifiers are handled through the Kotlin service layer and migration-provided columns.

5.7 `invoices`

Sales invoices and e-invoice tracking.

Key fields:

- `id`
- `organization_id`
- `customer_id`
- `invoice_number`
- dates: `invoice_date`, `due_date`, `sent_at`, `viewed_at`, `paid_at`

- money fields: `currency_code`, `exchange_rate`, `subtotal`, `tax_amount`, `discount_amount`, `total_amount`, `base_amount`
- `status`
- `notes`, `terms`, `pdf_url`
- e-invoice fields: `is_reverse_charge`, `sef_id`, `sef_document_id`, `sef_status`, `sef_submitted_at`, `sef_accepted_at`
- `created_by`
- lifecycle fields: `created_at`, `updated_at`, `version`, `deleted_at`

Column summary:

Column group	Type/constraint summary
Identity/scope	`id uuid pk`, `organization_id uuid fk`, `customer_id uuid fk`
Numbering/dates	invoice number, invoice/due dates, send/view/pay timestamps
Amounts	subtotal, tax, discount, total, base amount as numeric money values
Status	status text/enum-backed by migration history
E-invoice	reverse-charge flag and SEF IDs/status/timestamps
Ownership/lifecycle	creator user, timestamps, `version`, `deleted_at soft-delete`

Notes:

- Invoice status enum support was added by V18.
- Serbia SEF integration fields are present on the invoice table.

5.8 `invoice_items`

Line items for invoices.

Key fields:

- `id`
- `invoice_id`
- `line_number`
- `description`
- `quantity`
- `unit_price`
- `tax_rate`
- `vat_exempt`
- `line_total`
- `account_id`

- `created_at`
- `version`
- `deleted_at`

5.9 `recurring_invoices`

Recurring invoice templates/schedules.

Key fields:

- `id`
- `organization_id`
- `contact_id`
- `frequency`
- `next_issue_date`
- `day_of_month`
- `currency_code`
- `notes`
- `is_active`
- `template_data`
- `created_at`, `updated_at`

5.10 `expenses`

Purchase/expense records.

Key fields:

- `id`
- `organization_id`
- `vendor_id`
- `expense_number`
- `expense_date`
- money fields: `currency_code`, `exchange_rate`, `amount`, `base_amount`, `tax_amount`
- `category`
- `payment_method`
- `account_id`
- `description`
- `receipt_url`
- `status`
- approval/payment fields: `approved_by`, `approved_at`, `paid_at`
- `created_by`
- lifecycle fields: `created_at`, `updated_at`, `version`, `deleted_at`

Column summary:

Column group	Type/constraint summary
Identity/scope	`id uuid pk`, `organization_id uuid fk`, `vendor_id uuid fk`
Numbering/date	expense number and expense date
Amounts	amount, base amount, tax amount, currency, exchange rate
Classification	category, payment method, expense ledger account
Approval/payment	status, approver, approved timestamp, paid timestamp
Evidence/lifecycle	receipt URL, creator, timestamps, `version`, `deleted_at soft-delete`

Notes:

- Expense status enum support was added by V19.

5.11 `transactions`

General ledger transactions.

Key fields:

- `id`
- `organization_id`
- `transaction_date`
- `description`
- `debit_account_id`
- `credit_account_id`
- money fields: `amount`, `currency_code`, `exchange_rate`, `base_amount`
- source reference: `reference_type`, `reference_id`
- lock/reconciliation fields: `locked`, `locked_at`, `reconciled`, `reconciled_at`
- `notes`
- `created_by`
- `created_at`
- `version`
- `deleted_at`

Column summary:

Column group	Type/constraint summary
Identity/scope	`id uuid pk`, `organization_id uuid fk`
Double-entry legs	debit account FK, credit account FK
Amounts	amount, base amount, currency, exchange rate

Column group	Type/constraint summary
Source reference	reference type/id links back to invoices, expenses, or manual entries
Controls	lock and reconciliation flags/timestamps
Lifecycle	creator, `created_at`, `version`, `deleted_at soft-delete`

5.12 `bank_accounts`

Bank accounts linked to ledger accounts.

Key fields:

- `id`
- `organization_id`
- `account_id`
- `bank_name`
- `account_number`
- `iban`
- `currency_code`
- `current_balance`
- `is_active`
- lifecycle fields: `created_at`, `updated_at`, `version`, `deleted_at`

5.13 `bank_transactions`

Imported or entered bank movements.

Key fields:

- `id`
- `bank_account_id`
- `transaction_date`
- `amount`
- `description`
- `reference`
- `reconciled`
- `matched_transaction_id`
- `created_at`
- `version`
- `deleted_at`

5.14 `currencies`

Currency reference data.

Key fields:

- `code`
- `name`
- `symbol`
- `decimal_places`
- `is_active`
- `created_at`
- `version`

5.15 `exchange_rates`

Foreign-exchange rates.

Key fields:

- `id`
- `base_currency`
- `target_currency`
- `rate`
- `effective_date`
- `source`
- `last_updated`
- `version`
- `deleted_at`

5.16 `logged_actions`

Audit table populated by database/application audit paths.

Column summary:

Column group	Type/constraint summary
Identity	`event_id` primary identifier
Target	schema/table names
Actor/time	user ID, timestamp, client IP, application name
Change payload	action, row data, changed fields, query text

Key fields:

- `event_id`

- `schema_name`
- `table_name`
- `user_id`
- `action_timestamp`
- `action`
- `row_data`
- `changed_fields`
- `query`
- `client_ip`
- `application_name`

Notes:

- V28 widened the `action` column.
- V30+ migrations add auth/RLS-related grants and helper functions.

5.17 `chat_conversations`

AI assistant conversation storage.

Key fields:

- `id`
- `user_id`
- `organization_id`
- `messages`
- `updated_at`
- `version`
- `deleted_at`

5.18 `beta_interests`

Public/beta interest capture.

Key fields:

- `id`
- `email`
- `company_size`
- `use_case`
- `source`
- `created_at`
- `version`

5.19 `leads`

Lead capture records from public/landing flows.

Key fields:

- `id`
- `name`
- `email`
- `company`
- `phone`
- `country`
- `message`
- `lead_source`
- `ip`
- `user_agent`
- `status`
- `created_at`

5.20 `stripe_webhook_events`

Payment provider webhook idempotency/audit log.

Key fields:

- `id`
- `event_type`
- `organization_id`
- `payload`
- `processed_at`
- `error`

5.21 `sef_webhook_events`

SEF status webhook idempotency/audit log added by V36. Because SEF does not expose a documented immutable event ID, the API computes a SHA-256 idempotency key from the parsed SEF payload and raw body before calling `SefService.handleWebhook()`.

Key fields:

- `id` — SHA-256 idempotency key; primary key
- `sef_invoice_id`
- `status`
- `status_date`

- `payload`
- `processing_status` — `processing`, `processed`, or `failed`
- `processed_at`
- `error`
- `created_at`

Notes:

- Duplicate webhook deliveries with the same idempotency key return `200` with `duplicate=true` and are not reprocessed once processing is in progress or complete.
- Failed events are marked `failed`; a later duplicate delivery can retry processing.
- Signature verification still happens first via `X-Sef-Signature` / `SEF_WEBHOOK_SECRET`.

5.22 `adapter_config`

Per-market integration adapter toggles.

Key fields:

- `id`
- `market`
- `adapter_type`
- `adapter_name`
- `enabled`
- `reason`
- `updated_at`
- `updated_by`

Notes:

- Added by V25.
- Used to control market adapters such as e-invoice integrations.

5.23 `schema_version`

Legacy/internal schema marker table mapped by Exposed.

Key fields:

- `version`
- `applied_at`
- `description`

Notes:

- Flyway remains the migration authority. This table is not a replacement for Flyway history.

---

6. Cross-Cutting Conventions

UUID identifiers

Most business tables use UUID primary keys. Public API paths expose UUID strings for resource identifiers.

Soft deletion

Several tenant/business tables include `deleted_at`. Application queries should exclude soft-deleted rows unless a route is explicitly designed for archive/audit use.

Optimistic version field

Many tables include a `version` field. Preserve it when adding update paths and migrations.

Money

Money columns are stored as decimal/numeric values with explicit currency fields. `base_amount` fields support organization base-currency reporting.

Country and market support

Market-specific support currently includes HR/RS/BA concepts across country constraints, tax rates, chart-of-accounts migrations, SEF fields, and adapter configuration.

---

7. Operational Procedures

Add a table or column

1. Create a new Flyway migration with the next version.
2. Add or update the corresponding Exposed mapping in `Tables.kt`.
3. Update services/routes/tests that use the new field.
4. Update OpenAPI and backend docs if API shape changes.

5. Run Flyway validation/migration in the intended environment.
6. Capture evidence for MC/PR review.

Change an existing applied migration

Do not edit it. Instead:

1. Create a new forward migration.
2. Explain the compatibility path in the PR/MC evidence.
3. Validate on a non-production target before promotion.

Repair Flyway metadata drift

Only perform repair after all of these are captured:

1. Target identity: project, instance, database, environment.
2. Flyway validate output showing exact drift.
3. Schema checks proving the live schema matches expected intent.
4. Written runbook and abort conditions.
5. Repair transcript.
6. Post-repair validate/migrate/info output.
7. Deployment or smoke evidence if the drift blocked CI/CD.

MC #101509 is the reference example for this flow.

---

8. Validation Checklist

Before marking database documentation current:

- `Tables.kt` table inventory reviewed.
- Flyway migration directory reviewed.
- No stale deployment assumptions remain in this document.
- No legacy ORM workflow is presented as active.
- OpenAPI/API docs updated when endpoint shapes changed.
- Environment-specific migration claims cite evidence.

---

9. Index and Performance Strategy

The exact index inventory is migration-defined and should be inspected with `psql` against the target database when diagnosing query plans. The application design depends on these index

principles:

Query family	Required access pattern
Tenant lists	composite lookup by `organization_id` plus status/date/name as applicable
Invoice lists	organization + customer/status/date ordering
Expense lists	organization + vendor/status/date ordering
Ledger reports	organization + transaction date range; debit/credit account joins
Bank reconciliation	bank account + reconciliation status + transaction date
Auth	user email lookup and refresh-token `jti`/user lookup
Audit	table/action/time filtering and user/time filtering

Performance targets for product-facing paths:

- Tenant-scoped list endpoints should avoid full-table scans across organizations.
- Month/quarter report queries should be bounded by organization and date range.
- Background reconciliation/export jobs may use broader scans, but should be batchable and observable.
- Any new high-cardinality field used in filters should include an index decision in the migration PR.

When adding an index:

1. Add it in a new Flyway migration.
2. Explain the route/report it supports.
3. Verify with `EXPLAIN` or a representative query when data volume makes the risk material.

---

10. Audit Log Scaling and Retention

`logged_actions` can grow faster than ordinary tenant tables. Current documentation stance:

- The active schema keeps audit rows in PostgreSQL and records actor, target table, action, row data, changed fields, query text, client IP, and application name.
- V28 widened the action field to support current action labels.
- No partitioning migration is currently documented as applied in `Tables.kt`/Flyway source of truth.

Future scaling decision:

- If audit volume threatens report/API latency or storage budgets, introduce an explicit Flyway migration for partitioning or archival.
- The migration must include retention policy, query impact, backfill plan, and restore/audit requirements.
- Until that migration exists, do not describe partitioning as active behaviour.

---

11. Known Follow-Ups

- Keep `docs/backend/openapi.yaml` aligned with implemented Ktor routes.
- Keep `docs/backend/API-REFERENCE.md` aligned with OpenAPI.
- Keep deployment docs aligned with GCP Cloud Run and Cloud SQL reality.
- Consider generating a schema snapshot from a migrated Cloud SQL-compatible database for future reviews.

Revision #10
Created 2026-02-24 22:50:54 UTC by John
Updated 2026-06-07 19:42:52 UTC by John