API Reference

Bilko API Reference

> **Status:** SPECIFICATION (backend not implemented) **Base URL:** `http://localhost:4000/api/v1` (development) **Production URL:** `https://api.bilko.io/api/v1` **Last updated:** 2026-02-20

Purpose

This document is the implementation contract for Bilko's backend. All ~35 endpoints are specified with:

- HTTP method + path
- Authentication requirements
- Request/response TypeScript interfaces
- Query parameters
- Error responses
- Example requests/responses

CRITICAL: Backend is NOT BUILT. This is the spec that apps/api/ MUST implement.

Table of Contents

7. Bank Accounts (4 endpoints)
8. Reports (7 endpoints)
9. Chart of Accounts (3 endpoints)
10. Transactions (2 endpoints)
11. Settings (2 endpoints)
12. Currencies (2 endpoints)

Total: 50 endpoints

API Architecture Overview

```
graph LR
    subgraph CLIENT [Client]
        FE[Next.js Frontend\nbilko.io:3000]
    end

    subgraph API [Express API — api.bilko.io:4000]
        AUTH_R[/auth/*\nPublic]
        ORG_R[/organization\nAll roles]
        USR_R[/users/*\nowner, admin]
        CON_R[/contacts/*\nAll roles]
        INV_R[/invoices/*\nAll roles]
        EXP_R[/expenses/*\nAll roles]
        BANK_R[/bank-accounts/*\nAll roles]
        RPT_R[/reports/*\nAll roles]
        ACC_R[/accounts/*\nAll roles]
        TXN_R[/transactions/*\nAll roles]
        SET_R[/settings/*\nowner, admin]
        CUR_R[/currencies\nAll roles]
    end

    FE -->|Bearer token\nin Authorization header| API
    FE -->|refreshToken\nhttpOnly cookie| AUTH_R

    style AUTH_R fill:#e2e8f0,color:#000
    style FE fill:#00E5A0,color:#000
```

Global Response Patterns

Pagination

All list endpoints support pagination:

```
interface PaginatedResponse<T> {
  data: T[]
  meta: {
    total: number        // Total records
    page: number         // Current page (1-indexed)
    perPage: number      // Records per page
    totalPages: number   // Total pages
  }
}
```

Query parameters:

- `page` (default: 1)
- `perPage` (default: 20, max: 100)
- `sort` (field name, default varies by endpoint)
- `order` (`asc` or `desc`, default: `desc`)

Error Responses

```
interface ApiError {
  error: string                         // Human-readable error message
  code: string                          // Machine-readable error code
  details?: Record<string, string[]>    // Field-level validation errors
}
```

HTTP Status Codes:

- `400 Bad Request` — Invalid request body/params
- `401 Unauthorized` — Missing or invalid auth token
- `403 Forbidden` — User lacks required role
- `404 Not Found` — Resource does not exist
- `422 Unprocessable Entity` — Validation failed
- `500 Internal Server Error` — Server error

---

1. Authentication

POST /api/v1/auth/register

Create new organization and owner user.

Auth: None **Rate limit:** 5 req/min

Request:

```
interface RegisterRequest {
  // Organization
  organizationName: string
  country: 'RS' | 'BA' | 'HR'        // Serbia, BiH, Croatia
  baseCurrency: 'EUR' | 'RSD' | 'BAM' | 'HRK'
  language: 'sr' | 'bs' | 'hr'
  registrationNumber?: string         // Company tax ID
  vatNumber?: string

  // User
  email: string                       // Must be unique
  password: string                    // Min 8 chars, 1 upper, 1 lower, 1 number
  fullName: string
}
```

Response (201):

```
interface RegisterResponse {
  user: {
    id: string
    email: string
    fullName: string
    role: 'owner'
  }
  organization: {
    id: string
    name: string
    country: string
    baseCurrency: string
  }
```

```
  tokens: {
    accessToken: string      // JWT, expires in 15 min
    refreshToken: string     // Expires in 7 days
  }
}
```

Errors:

- `400` — Email already exists
- `422` — Validation failed (weak password, invalid country, etc.)

POST /api/v1/auth/login

Authenticate with email + password.

Auth: None **Rate limit:** 5 req/min

Request:

```
interface LoginRequest {
  email: string
  password: string
  rememberMe?: boolean     // If true, refreshToken expires in 30 days
}
```

Response (200):

```
interface LoginResponse {
  user: {
    id: string
    email: string
    fullName: string
    role: 'owner' | 'admin' | 'accountant' | 'viewer'
    organizationId: string
    organizationName: string
  }
  tokens: {
    accessToken: string      // JWT, expires in 15 min
    refreshToken: string     // httpOnly cookie
  }
}
```

Errors:

- `401` — Invalid credentials
- `403` — Account disabled or requires 2FA

POST /api/v1/auth/refresh

Get new access token using refresh token.

Auth: Refresh token (httpOnly cookie) **Rate limit:** 100 req/min

Request: None (uses cookie)

Response (200):

```
interface RefreshResponse {
  accessToken: string
}
```

Errors:

- `401` — Invalid or expired refresh token

POST /api/v1/auth/logout

Invalidate refresh token.

Auth: Bearer token **Rate limit:** 100 req/min

Request: None

Response (204): No content

GET /api/v1/auth/me

Get current user info.

Auth: Bearer token **Rate limit:** 100 req/min

Response (200):

```
interface CurrentUser {
  id: string
  email: string
  fullName: string
  role: 'owner' | 'admin' | 'accountant' | 'viewer'
  twoFactorEnabled: boolean
  lastLoginAt: string | null
  organization: {
    id: string
    name: string
    country: string
    baseCurrency: string
    language: string
  }
}
```

---

2. Organization

GET /api/v1/organization

Get organization details.

Auth: Bearer token **Roles:** All **Rate limit:** 100 req/min

Response (200):

```
interface Organization {
  id: string
  name: string
  registrationNumber: string | null
  vatNumber: string | null
  baseCurrency: string
  country: string
  language: string
  fiscalYearStart: string    // ISO date, e.g., "2026-01-01"
  createdAt: string
  updatedAt: string
}
```

PUT /api/v1/organization

Update organization details.

Auth: Bearer token **Roles:** owner, admin **Rate limit:** 10 req/min

Request:

```
interface UpdateOrganizationRequest {
  name?: string
  registrationNumber?: string
  vatNumber?: string
  baseCurrency?: 'EUR' | 'RSD' | 'BAM' | 'HRK'
  language?: 'sr' | 'bs' | 'hr'
  fiscalYearStart?: string    // ISO date
}
```

Response (200): Organization object (same as GET)

Errors:

- `422` — Validation failed (invalid currency code, etc.)

---

3. Users

GET /api/v1/users

List all users in organization.

Auth: Bearer token **Roles:** owner, admin **Rate limit:** 100 req/min

Query:

- `role` (filter by role)

Response (200):

```
interface UserListResponse {
  data: Array<{
```

```
    id: string
    email: string
    fullName: string
    role: 'owner' | 'admin' | 'accountant' | 'viewer'
    twoFactorEnabled: boolean
    lastLoginAt: string | null
    createdAt: string
  }>
}
```

POST /api/v1/users/invite

Invite new user to organization.

Auth: Bearer token **Roles:** owner, admin **Rate limit:** 10 req/min

Request:

```
interface InviteUserRequest {
  email: string
  fullName: string
  role: 'admin' | 'accountant' | 'viewer'    // Cannot create 'owner'
}
```

Response (201):

```
interface InviteUserResponse {
  user: {
    id: string
    email: string
    fullName: string
    role: string
  }
  inviteLink: string    // One-time setup link, expires in 7 days
}
```

Errors:

- `400` — Email already exists in organization
- `422` — Invalid role

PUT /api/v1/users/:id/role

Change user role.

Auth: Bearer token **Roles:** owner **Rate limit:** 10 req/min

Request:

```
interface ChangeRoleRequest {
  role: 'admin' | 'accountant' | 'viewer'
}
```

Response (200): User object

Errors:

- `403` — Cannot change owner role or demote yourself
- `404` — User not found

DELETE /api/v1/users/:id

Remove user from organization.

Auth: Bearer token **Roles:** owner **Rate limit:** 10 req/min

Response (204): No content

Errors:

- `403` — Cannot delete owner or yourself
- `404` — User not found

---

4. Contacts

GET /api/v1/contacts

List contacts (customers/vendors).

Auth: Bearer token **Roles:** All **Rate limit:** 100 req/min

Query:

- `type` (`customer`, `vendor`, `both`)
- `page`, `perPage`, `sort`, `order`

Response (200):

```
type ContactListResponse = PaginatedResponse<Contact>


interface Contact {
  id: string
  type: 'customer' | 'vendor' | 'both'
  name: string
  email: string | null
  phone: string | null
  registrationNumber: string | null
  vatNumber: string | null
  addressLine1: string | null
  addressLine2: string | null
  city: string | null
  postalCode: string | null
  country: string | null
  currencyCode: string
  paymentTerms: number           // Days
  isActive: boolean
  createdAt: string
  updatedAt: string
}
```

POST /api/v1/contacts

Create new contact.

Auth: Bearer token **Roles:** owner, admin, accountant **Rate limit:** 50 req/min

Request:

```
interface CreateContactRequest {
  type: 'customer' | 'vendor' | 'both'
  name: string
```

```
  email?: string
  phone?: string
  registrationNumber?: string
  vatNumber?: string
  addressLine1?: string
  addressLine2?: string
  city?: string
  postalCode?: string
  country?: string               // ISO 3166-1 alpha-2 (e.g., 'RS')
  currencyCode?: string          // ISO 4217 (default: org baseCurrency)
  paymentTerms?: number          // Default: 30 days
  notes?: string
}
```

Response (201): Contact object

Errors:

- `422` — Validation failed (invalid country code, currency code, etc.)

GET /api/v1/contacts/:id

Get contact details.

Auth: Bearer token **Roles:** All **Rate limit:** 100 req/min

Response (200): Contact object + notes field

PUT /api/v1/contacts/:id

Update contact.

Auth: Bearer token **Roles:** owner, admin, accountant **Rate limit:** 50 req/min

Request: Same as CreateContactRequest (all fields optional)

Response (200): Contact object

DELETE /api/v1/contacts/:id

Soft-delete contact (sets isActive = false).

Auth: Bearer token **Roles:** owner, admin **Rate limit:** 10 req/min

Response (204): No content

Errors:

- `400` — Contact has active invoices or expenses

---

5. Invoices

GET /api/v1/invoices

List invoices.

Auth: Bearer token **Roles:** All **Rate limit:** 100 req/min

Query:

- `status` (`draft`, `sent`, `viewed`, `paid`, `overdue`, `cancelled`)
- `customerId` (UUID)
- `fromDate`, `toDate` (ISO dates)
- `page`, `perPage`, `sort`, `order`

Response (200):

```
type InvoiceListResponse = PaginatedResponse<InvoiceSummary>


interface InvoiceSummary {
  id: string
  invoiceNumber: string
  customerId: string
  customerName: string
  invoiceDate: string
  dueDate: string
  currencyCode: string
  totalAmount: string         // Decimal as string, e.g., "125000.0000"
  status: 'draft' | 'sent' | 'viewed' | 'paid' | 'overdue' | 'cancelled'
  createdAt: string
}
```

Invoice Creation — Full Sequence

```
sequenceDiagram
    participant FE as Frontend
    participant MW as Middleware Stack\n(auth, roleGuard, validate)
    participant H as Invoice Handler
    participant DB as PostgreSQL\n(Prisma)
    participant EX as Exchange Rate\nService

    FE->>MW: POST /api/v1/invoices\nAuthorization: Bearer {accessToken}
    MW->>MW: authGuard: verify JWT\nAttach req.user {id, role, orgId}
    MW->>MW: roleGuard: check owner/admin/accountant
    MW->>MW: validate(createInvoiceSchema)\ncustomerId UUID, dates, items[]

    MW->>H: Validated request
    H->>DB: Find Contact by customerId\nwhere orgId matches
    DB-->>H: Contact { email, currencyCode }

    H->>EX: getExchangeRate(invoiceCurrency, orgBaseCurrency, invoiceDate)
    EX-->>H: rate (locked at invoiceDate — NEVER changes)

    H->>H: Calculate:\nlineTotal = qty × unitPrice\ntaxAmount = SUM(lineTotal × taxRate/100)\ntotalAmount = subtotal + taxAmount - discount\nbaseAmount = totalAmount × exchangeRate

    H->>DB: BEGIN TRANSACTION\nGenerate invoiceNumber INV-YYYY-NNN\nINSERT Invoice { status: draft }\nINSERT InvoiceItems[]

    DB-->>H: Invoice created
    H->>DB: INSERT LoggedAction\n{ action: INSERT, tableName: Invoice }
    DB-->>H: Logged

    H->>FE: 201 Created\n{ id, invoiceNumber, status: draft, items, totals }
```

POST /api/v1/invoices

Create invoice.

Auth: Bearer token **Roles:** owner, admin, accountant **Rate limit:** 50 req/min

Request:

```
interface CreateInvoiceRequest {
  customerId: string
  invoiceDate: string           // ISO date
  dueDate: string               // ISO date
  currencyCode?: string         // Default: customer's currency
  items: Array<{
    description: string
    quantity: number            // Decimal as number
    unitPrice: number           // Decimal as number
    taxRate: number             // Percentage, e.g., 20 for 20%
    accountId?: string          // Revenue account
  }>
  notes?: string
  terms?: string
}
```

Response (201):

```
interface Invoice {
  id: string
  invoiceNumber: string         // Auto-generated
  customerId: string
  customerName: string
  invoiceDate: string
  dueDate: string
  currencyCode: string
  exchangeRate: string          // Decimal as string
  subtotal: string
  taxAmount: string
  discountAmount: string
  totalAmount: string
  baseAmount: string            // Converted to org baseCurrency
  status: 'draft'
  items: Array<{
    id: string
    lineNumber: number
    description: string
```

```
    quantity: string
    unitPrice: string
    taxRate: string
    lineTotal: string
    accountId: string | null
  }>
  notes: string | null
  terms: string | null
  pdfUrl: string | null
  createdBy: string
  createdAt: string
  updatedAt: string
}
```

Errors:

- `404` — Customer not found
- `422` — Validation failed (invalid date, negative amount, etc.)

GET /api/v1/invoices/:id

Get invoice details.

Auth: Bearer token **Roles:** All **Rate limit:** 100 req/min

Response (200): Invoice object (same as POST response)

PUT /api/v1/invoices/:id

Update invoice (draft only).

Auth: Bearer token **Roles:** owner, admin, accountant **Rate limit:** 50 req/min

Request: Same as CreateInvoiceRequest

Response (200): Invoice object

Errors:

- `400` — Invoice is not in draft status

Invoice Status Transition — Send Flow

```
sequenceDiagram
    participant FE as Frontend
    participant API as Bilko API
    participant PDF as Puppeteer\nPDF Service
    participant R2 as Cloudflare R2
    participant SG as SendGrid
    participant DB as PostgreSQL

    FE->>API: PATCH /invoices/:id/status\n{ action: "send" }
    API->>DB: Fetch Invoice with items, customer, org
    DB-->>API: Invoice (must be status=draft)
    API->>PDF: generateInvoicePDF(invoice data)
    PDF-->>API: PDF Buffer

    API->>R2: PUT invoices/{orgId}/INV-2026-001.pdf
    R2-->>API: pdfUrl stored

    API->>DB: BEGIN TRANSACTION
    API->>DB: INSERT Transaction {\n  DR: Accounts Receivable (1200)\n  CR: Revenue (4000)\n  amount: invoice.totalAmount\n  referenceType: 'invoice'\n}
    API->>DB: UPDATE Invoice SET\n  status='sent', sentAt=now()\n  pdfUrl=url

    API->>SG: sendEmail({\n  to: customer.email,\n  subject: "Invoice INV-2026-001 from Org",\n  html: template,\n  attachment: pdf\n})
    SG-->>API: { messageId }

    DB-->>API: COMMIT

    API->>FE: 200 { status: sent, sentAt, pdfUrl }

    Note over API: If SendGrid fails:\nKeep invoice as draft\nAdd note: "Email delivery failed"\nAlert admin via Slack
```

PATCH /api/v1/invoices/:id/status

Change invoice status.

Auth: Bearer token **Roles:** owner, admin, accountant **Rate limit:** 50 req/min

Request:

```
interface ChangeInvoiceStatusRequest {
  action: 'send' | 'mark-paid' | 'cancel'
  paidAt?: string               // Required if action = 'mark-paid'
}
```

Response (200): Invoice object

Business logic:

- `send`: draft → sent (generates PDF, sends email via SendGrid)
- `mark-paid`: sent/viewed → paid (creates Transaction: debit BankAccount, credit AccountsReceivable)
- `cancel`: any → cancelled (reverses Transaction if paid)

Errors:

- `400` — Invalid status transition

GET /api/v1/invoices/:id/pdf

Get invoice PDF.

Auth: Bearer token **Roles:** All **Rate limit:** 100 req/min

Response (200):

- Content-Type: application/pdf
- Content-Disposition: attachment; filename="INV-2026-001.pdf"

Errors:

- `404` — Invoice or PDF not found

POST /api/v1/invoices/:id/send

Send invoice email to customer.

Auth: Bearer token **Roles:** owner, admin, accountant **Rate limit:** 10 req/min

Request:

```
interface SendInvoiceRequest {
  to?: string                   // Override customer email
  cc?: string[]
  subject?: string              // Override default subject
  message?: string              // Custom message
}
```

Response (200):

```
interface SendInvoiceResponse {
  sentAt: string
  sentTo: string
  emailId: string               // SendGrid message ID
}
```

Errors:

- `400` — Customer has no email
- `500` — SendGrid error

---

6. Expenses

GET /api/v1/expenses

List expenses.

Auth: Bearer token **Roles:** All **Rate limit:** 100 req/min

Query:

- `status` (`pending`, `approved`, `paid`, `rejected`)
- `category`
- `vendorId`
- `fromDate`, `toDate`
- `page`, `perPage`, `sort`, `order`

Response (200):

```
type ExpenseListResponse = PaginatedResponse<ExpenseSummary>
```

```
interface ExpenseSummary {
  id: string
  expenseNumber: string
  vendorId: string | null
  vendorName: string | null
  expenseDate: string
  category: string
  amount: string
  currencyCode: string
  status: 'pending' | 'approved' | 'paid' | 'rejected'
  receiptUrl: string | null
  createdAt: string
}
```

POST /api/v1/expenses

Create expense.

Auth: Bearer token **Roles:** owner, admin, accountant **Rate limit:** 50 req/min

Request:

```
interface CreateExpenseRequest {
  vendorId?: string
  expenseDate: string
  category: string              // Free text or predefined categories
  amount: number
  currencyCode?: string         // Default: org baseCurrency
  taxAmount?: number
  paymentMethod?: string        // 'cash', 'card', 'bank_transfer', etc.
  accountId?: string            // Expense account
  description?: string
  receiptFile?: File            // Multipart form upload (max 10MB)
}
```

Response (201):

```
interface Expense {
  id: string
  expenseNumber: string         // Auto-generated
```

```
  vendorId: string | null
  vendorName: string | null
  expenseDate: string
  category: string
  currencyCode: string
  exchangeRate: string
  amount: string
  baseAmount: string
  taxAmount: string
  paymentMethod: string | null
  accountId: string | null
  description: string | null
  receiptUrl: string | null     // Cloudflare R2 URL
  status: 'pending'
  createdBy: string
  createdAt: string
  updatedAt: string
}
```

Errors:

- `422` — Validation failed (negative amount, invalid date, etc.)
- `413` — File too large

GET /api/v1/expenses/:id

Get expense details.

Auth: Bearer token **Roles:** All **Rate limit:** 100 req/min

Response (200): Expense object

PUT /api/v1/expenses/:id

Update expense (pending only).

Auth: Bearer token **Roles:** owner, admin, accountant **Rate limit:** 50 req/min

Request: Same as CreateExpenseRequest

Response (200): Expense object

Errors:

- `400` — Expense is not pending

Expense Approval — Full Sequence

```
sequenceDiagram
    participant FE as Frontend\n(admin/owner)
    participant MW as Middleware Stack
    participant H as Expense Handler
    participant DB as PostgreSQL

    FE->>MW: PATCH /api/v1/expenses/:id/approve\nAuthorization: Bearer {accessToken}
    MW->>MW: authGuard: verify JWT
    MW->>MW: roleGuard: owner or admin ONLY\n(accountant CANNOT approve)

    MW->>H: Request passes
    H->>DB: Find Expense by id\nwhere organizationId = req.orgId
    DB-->>H: Expense { status: pending, amount, accountId }

    H->>H: Validate: status must be 'pending'\nIf not → 400 Bad Request

    H->>DB: BEGIN TRANSACTION

    H->>DB: Find ExpenseAccount\n(expense.accountId or default 5xxx)
    H->>DB: Find AccountsPayable account\n(2110 or configured account)

    H->>DB: INSERT Transaction {\n  debitAccountId: expenseAccountId,\n  creditAccountId: accountsPayableId,\n  amount: expense.amount,\n  referenceType: 'expense',\n  referenceId: expense.id\n}

    H->>DB: UPDATE Expense SET\n  status='approved',\n  approvedBy=req.user.id,\n  approvedAt=now()

    H->>DB: INSERT LoggedAction

    DB-->>H: COMMIT
```

```
    H->>FE: 200 OK\n{ id, status: approved, approvedBy, approvedAt }
```

PATCH /api/v1/expenses/:id/approve

Approve expense.

Auth: Bearer token **Roles:** owner, admin **Rate limit:** 50 req/min

Response (200): Expense object (status = approved)

Business logic:

- Creates Transaction: debit ExpenseAccount, credit AccountsPayable

Errors:

- `400` — Expense already approved/paid/rejected

DELETE /api/v1/expenses/:id

Delete expense (pending only).

Auth: Bearer token **Roles:** owner, admin **Rate limit:** 10 req/min

Response (204): No content

Errors:

- `400` — Expense is not pending

---

7. Bank Accounts

GET /api/v1/bank-accounts

List bank accounts.

Auth: Bearer token **Roles:** All **Rate limit:** 100 req/min

Response (200):

```
interface BankAccountListResponse {
  data: Array<{
    id: string
    accountId: string           // GL account ID
    accountCode: string         // GL account code
    bankName: string
    accountNumber: string | null
    iban: string | null
    currencyCode: string
    currentBalance: string
    isActive: boolean
    createdAt: string
    updatedAt: string
  }>
}
```

POST /api/v1/bank-accounts

Create bank account.

Auth: Bearer token **Roles:** owner, admin **Rate limit:** 10 req/min

Request:

```
interface CreateBankAccountRequest {
  accountId: string             // Must be Asset account
  bankName: string
  accountNumber?: string
  iban?: string
  currencyCode: string
  currentBalance?: number       // Default: 0
}
```

Response (201): BankAccount object

Errors:

- `404` — Account not found
- `422` — Account is not Asset type

GET /api/v1/bank-accounts/:id/transactions

Get bank transactions.

Auth: Bearer token **Roles:** All **Rate limit:** 100 req/min

Query:

- `fromDate`, `toDate`
- `reconciled` (true/false)
- `page`, `perPage`, `sort`, `order`

Response (200):

```
type BankTransactionListResponse = PaginatedResponse<BankTransaction>


interface BankTransaction {
  id: string
  transactionDate: string
  amount: string                // Positive = credit, negative = debit
  description: string | null
  reference: string | null
  reconciled: boolean
  matchedTransactionId: string | null
  createdAt: string
}
```

POST /api/v1/bank-accounts/:id/import

Import bank statement (CSV).

Auth: Bearer token **Roles:** owner, admin, accountant **Rate limit:** 10 req/min

Request:

- Multipart form: `file` (CSV, max 5MB)

CSV format:

```
Date,Description,Amount,Reference
2026-02-19,"Payment from customer",3500.00,INV-2026-002
2026-02-18,"AWS Invoice",-850.00,
```

Response (200):

```
interface ImportStatementResponse {
  imported: number
  duplicates: number
  errors: Array<{
    line: number
    error: string
  }>
}
```

Errors:

- `422` — Invalid CSV format
- `413` — File too large

Bank Reconciliation — Full Sequence

```
sequenceDiagram
    participant FE as Frontend
    participant API as Bilko API
    participant DB as PostgreSQL

    Note over FE,DB: Step 1 — Import Bank Statement
    FE->>API: POST /bank-accounts/:id/import\n[multipart: CSV file, max 5MB]
    API->>API: Parse CSV\nDate, Description, Amount, Reference
    API->>DB: INSERT BankTransaction[] records\n{ bankAccountId, transactionDate, amount,
reference }
    DB-->>API: Imported count
    API->>FE: 200 { imported: 45, duplicates: 2, errors: [] }

    Note over FE,DB: Step 2 — Auto-Match Suggestions
    FE->>API: GET /bank-accounts/:id/transactions?reconciled=false
    API->>DB: Fetch unreconciled BankTransactions
    DB-->>API: BankTransaction[]
    API->>DB: Fetch unreconciled GL Transactions\nfor same date range
    DB-->>API: Transaction[]
    API->>API: calculateMatchScore() for each pair\nAmount match +50\nDate match
+30/+20/+10\nReference match +20
```

```
    API->>FE: 200 { bankTransactions, suggestions[{ bankTxId, glTxId, score }] }

    Note over FE,DB: Step 3 — Confirm Reconciliation
    FE->>API: POST /bank-accounts/:id/reconcile\n{ bankTransactionId, transactionId }
    API->>DB: Find both records, verify same org
    API->>DB: UPDATE BankTransaction SET\n  reconciled=true\n  matchedTransactionId=glTxId
    API->>DB: UPDATE Transaction SET\n  reconciled=true\n  reconciledAt=now()
    DB-->>API: Both updated
    API->>FE: 200 { bankTransaction, transaction, confidence: 95 }
```

POST /api/v1/bank-accounts/:id/reconcile

Reconcile bank transactions with GL transactions.

Auth: Bearer token **Roles:** owner, admin, accountant **Rate limit:** 10 req/min

Request:

```
interface ReconcileRequest {
  bankTransactionId: string
  transactionId: string         // GL transaction ID
}
```

Response (200):

```
interface ReconcileResponse {
  bankTransaction: BankTransaction
  transaction: Transaction
  confidence: number            // 0-100 match score
}
```

Errors:

- `404` — Bank transaction or GL transaction not found
- `400` — Already reconciled

---

8. Reports

GET /api/v1/reports/dashboard

Get dashboard metrics.

Auth: Bearer token **Roles:** All **Rate limit:** 100 req/min

Response (200):

```
interface DashboardMetrics {
  cashBalance: string           // Total across all bank accounts (in baseCurrency)
  revenueMTD: string            // Month-to-date revenue
  unpaidInvoices: string        // Total unpaid invoices
  expensesMTD: string           // Month-to-date expenses
  profitMTD: string             // revenueMTD - expensesMTD
  cashFlowChange: number        // Percentage change from last month

  // Chart data
  monthlyPL: Array<{
    month: string
    revenue: string
    expenses: string
    profit: string
  }>

  receivablesAging: {
    current: string             // 0-30 days
    days30: string              // 31-60 days
    days60: string              // 61-90 days
    days90plus: string          // 90+ days
  }

  expensesByCategory: Array<{
    category: string
    amount: string
    currencyCode: string
  }>
}
```

GET /api/v1/reports/profit-loss

Profit & Loss statement.

Auth: Bearer token **Roles:** All **Rate limit:** 50 req/min

Query:

- `from` (ISO date, required)
- `to` (ISO date, required)

Response (200):

```
interface ProfitLossReport {
  period: {
    from: string
    to: string
  }
  baseCurrency: string

  revenue: {
    total: string
    accounts: Array<{
      accountCode: string
      accountName: string
      amount: string
    }>
  }

  expenses: {
    total: string
    accounts: Array<{
      accountCode: string
      accountName: string
      amount: string
    }>
  }

  netProfit: string             // revenue.total - expenses.total
}
```

GET /api/v1/reports/balance-sheet

Balance Sheet.

Auth: Bearer token **Roles:** All **Rate limit:** 50 req/min

Query:

- `date` (ISO date, default: today)

Response (200):

```
interface BalanceSheetReport {
  asOfDate: string
  baseCurrency: string

  assets: {
    total: string
    current: {
      total: string
      accounts: Array<AccountBalance>
    }
    fixed: {
      total: string
      accounts: Array<AccountBalance>
    }
  }

  liabilities: {
    total: string
    current: {
      total: string
      accounts: Array<AccountBalance>
    }
    longTerm: {
      total: string
      accounts: Array<AccountBalance>
    }
  }

  equity: {
    total: string
    accounts: Array<AccountBalance>
  }
}
```

```
interface AccountBalance {
  accountCode: string
  accountName: string
  balance: string
}
```

GET /api/v1/reports/cash-flow

Cash Flow statement.

Auth: Bearer token **Roles:** All **Rate limit:** 50 req/min

Query:

- `from`, `to` (ISO dates, required)

Response (200):

```
interface CashFlowReport {
  period: {
    from: string
    to: string
  }
  baseCurrency: string

  operating: {
    total: string
    items: Array<{
      description: string
      amount: string
    }>
  }

  investing: {
    total: string
    items: Array<{
      description: string
      amount: string
    }>
```

```
  }

  financing: {
    total: string
    items: Array<{
      description: string
      amount: string
    }>
  }

  netCashFlow: string
  openingBalance: string
  closingBalance: string
}
```

GET /api/v1/reports/vat

VAT/PDV report.

Auth: Bearer token **Roles:** All **Rate limit:** 50 req/min

Query:

- `from`, `to` (ISO dates, required)

Response (200):

```
interface VATReport {
  period: {
    from: string
    to: string
  }
  country: string               // Organization country

  outputVAT: {
    total: string
    invoices: Array<{
      invoiceNumber: string
      customerName: string
      invoiceDate: string
```

```
      baseAmount: string
      vatAmount: string
      vatRate: string
    }>
  }

  inputVAT: {
    total: string
    expenses: Array<{
      expenseNumber: string
      vendorName: string
      expenseDate: string
      baseAmount: string
      vatAmount: string
      vatRate: string
    }>
  }

  netVAT: string                // outputVAT.total - inputVAT.total

  reconciliationStatus: {
    allInvoicesPaid: boolean
    allExpensesApproved: boolean
    unmatchedTransactions: number
  }
}
```

GET /api/v1/reports/trial-balance

Trial Balance.

Auth: Bearer token **Roles:** All **Rate limit:** 50 req/min

Query:

- `date` (ISO date, default: today)

Response (200):

```
interface TrialBalanceReport {
  asOfDate: string
  baseCurrency: string

  accounts: Array<{
    accountCode: string
    accountName: string
    accountType: string
    debitTotal: string
    creditTotal: string
    balance: string
  }>

  totals: {
    debit: string
    credit: string
  }

  balanced: boolean             // totals.debit === totals.credit
}
```

---

9. Chart of Accounts

GET /api/v1/accounts

List chart of accounts.

Auth: Bearer token **Roles:** All **Rate limit:** 100 req/min

Query:

- `accountTypeId` (filter by type)
- `isActive` (true/false)

Response (200):

```
interface AccountListResponse {
  data: Array<{
    id: string
```

```
    code: string                // e.g., "1000", "4000"
    name: string                // e.g., "Bank Account EUR", "Revenue"
    accountTypeId: number
    accountTypeName: string     // Asset, Liability, Equity, Revenue, Expense
    normalBalance: 'debit' | 'credit'
    currencyCode: string
    parentAccountId: string | null
    parentAccountCode: string | null
    isActive: boolean
    currentBalance: string      // Calculated from transactions
    createdAt: string
    updatedAt: string
  }>
}
```

POST /api/v1/accounts

Create account.

Auth: Bearer token **Roles:** owner, admin **Rate limit:** 10 req/min

Request:

```
interface CreateAccountRequest {
  code: string                  // Must be unique within organization
  name: string
  accountTypeId: number         // 1-5 (Asset, Liability, Equity, Revenue, Expense)
  currencyCode?: string         // Default: org baseCurrency
  parentAccountId?: string      // For sub-accounts
}
```

Response (201): Account object

Errors:

- `400` — Code already exists
- `404` — Parent account not found
- `422` — Invalid account type

PUT /api/v1/accounts/:id

Update account.

Auth: Bearer token **Roles:** owner, admin **Rate limit:** 10 req/min

Request:

```
interface UpdateAccountRequest {
  name?: string
  isActive?: boolean
}
```

Response (200): Account object

Errors:

- `400` — Cannot deactivate account with transactions

---

10. Transactions

GET /api/v1/transactions

List general ledger transactions.

Auth: Bearer token **Roles:** All **Rate limit:** 100 req/min

Query:

- `fromDate`, `toDate`
- `accountId` (show transactions for specific account)
- `referenceType` (`invoice`, `expense`, `payment`, `manual`)
- `page`, `perPage`, `sort`, `order`

Response (200):

```
type TransactionListResponse = PaginatedResponse<Transaction>


interface Transaction {
  id: string
  transactionDate: string
```

```
  description: string
  debitAccountId: string
  debitAccountCode: string
  debitAccountName: string
  creditAccountId: string
  creditAccountCode: string
  creditAccountName: string
  amount: string
  currencyCode: string
  exchangeRate: string
  baseAmount: string
  referenceType: string | null
  referenceId: string | null
  locked: boolean
  reconciled: boolean
  createdBy: string
  createdAt: string
}
```

POST /api/v1/transactions

Create manual journal entry.

Auth: Bearer token **Roles:** owner, admin, accountant **Rate limit:** 20 req/min

Request:

```
interface CreateTransactionRequest {
  transactionDate: string
  description: string
  debitAccountId: string
  creditAccountId: string
  amount: number
  currencyCode?: string         // Default: org baseCurrency
  notes?: string
}
```

Response (201): Transaction object

Errors:

- `404` — Account not found
- `422` — Validation failed (debit = credit account, negative amount, etc.)

---

11. Settings

GET /api/v1/settings/tax-rates

Get tax rate configuration.

Auth: Bearer token **Roles:** All **Rate limit:** 100 req/min

Response (200):

```
interface TaxRatesResponse {
  country: string
  defaultVATRate: number        // e.g., 20 for Serbia, 17 for BiH
  rates: Array<{
    name: string                // "Standard", "Reduced", "Zero"
    rate: number
    description: string
  }>
}
```

PUT /api/v1/settings/tax-rates

Update tax rate configuration.

Auth: Bearer token **Roles:** owner, admin **Rate limit:** 10 req/min

Request:

```
interface UpdateTaxRatesRequest {
  defaultVATRate: number
  rates: Array<{
    name: string
    rate: number
    description: string
  }>
```

```
}
```

Response (200): TaxRatesResponse

---

12. Currencies

GET /api/v1/currencies

List supported currencies.

Auth: Bearer token **Roles:** All **Rate limit:** 100 req/min

Response (200):

```
interface CurrencyListResponse {
  data: Array<{
    code: string                // ISO 4217
    name: string
    symbol: string | null
    decimalPlaces: number
    isActive: boolean
  }>
}
```

GET /api/v1/exchange-rates

Get exchange rates.

Auth: Bearer token **Roles:** All **Rate limit:** 100 req/min

Query:

- `base` (currency code, required)
- `target` (currency code, required)
- `date` (ISO date, default: today)

Response (200):

```
interface ExchangeRateResponse {
  baseCurrency: string
  targetCurrency: string
  rate: string                  // Decimal as string
  effectiveDate: string
  source: string                // "ECB", "fixer.io", "manual"
  lastUpdated: string
}
```

Errors:

- `404` — No rate found for date (return nearest available)

Endpoint Summary Map

```
graph TD
    subgraph AUTH [Authentication — No auth required]
        A1[POST /auth/register]
        A2[POST /auth/login]
        A3[POST /auth/refresh]
        A4[POST /auth/logout]
        A5[GET /auth/me]
    end

    subgraph ORG [Organization]
        O1[GET /organization]
        O2[PUT /organization]
    end

    subgraph USR [Users]
        U1[GET /users]
        U2[POST /users/invite]
        U3[PUT /users/:id/role]
        U4[DELETE /users/:id]
    end

    subgraph CON [Contacts]
```

```
    C1[GET /contacts]
    C2[POST /contacts]
    C3[GET /contacts/:id]
    C4[PUT /contacts/:id]
    C5[DELETE /contacts/:id]
end

subgraph INV [Invoices]
    I1[GET /invoices]
    I2[POST /invoices]
    I3[GET /invoices/:id]
    I4[PUT /invoices/:id]
    I5[PATCH /invoices/:id/status]
    I6[GET /invoices/:id/pdf]
    I7[POST /invoices/:id/send]
end

subgraph EXP [Expenses]
    E1[GET /expenses]
    E2[POST /expenses]
    E3[GET /expenses/:id]
    E4[PUT /expenses/:id]
    E5[PATCH /expenses/:id/approve]
    E6[DELETE /expenses/:id]
end

subgraph BANK [Bank Accounts]
    B1[GET /bank-accounts]
    B2[POST /bank-accounts]
    B3[GET /bank-accounts/:id/transactions]
    B4[POST /bank-accounts/:id/import]
    B5[POST /bank-accounts/:id/reconcile]
end

subgraph RPT [Reports]
    R1[GET /reports/dashboard]
    R2[GET /reports/profit-loss]
    R3[GET /reports/balance-sheet]
    R4[GET /reports/cash-flow]
```

```
        R5[GET /reports/vat]
        R6[GET /reports/trial-balance]
    end

    subgraph MISC [Other]
        M1[GET /accounts]
        M2[POST /accounts]
        M3[PUT /accounts/:id]
        M4[GET /transactions]
        M5[POST /transactions]
        M6[GET /settings/tax-rates]
        M7[PUT /settings/tax-rates]
        M8[GET /currencies]
        M9[GET /exchange-rates]
    end
```

Implementation Notes

Request Validation

All requests validated with Zod schemas. Invalid requests return `422` with field-level errors.

Database Transactions

All write operations wrapped in database transactions. Rollback on error.

Audit Logging

All INSERT/UPDATE/DELETE captured in `LoggedAction` table via Prisma middleware.

Rate Limiting

- General: 100 req/min per user
- Auth: 5 req/min per IP
- Write ops: 10-50 req/min per user

File Uploads

- Max size: 10MB (receipts), 5MB (CSV)
- Allowed: PDF, PNG, JPG, CSV
- Storage: Cloudflare R2
- Virus scanning: ClamAV

CORS

- Allowed origins: `https://bilko.io`, `http://localhost:3000`
- Credentials: true (cookies)

Error Logging

- Sentry for production errors
- Winston for structured logs

Example Requests

Create Invoice

```
curl -X POST http://localhost:4000/api/v1/invoices \
  -H "Authorization: Bearer $TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "customerId": "550e8400-e29b-41d4-a716-446655440000",
    "invoiceDate": "2026-02-20",
    "dueDate": "2026-03-20",
    "items": [
      {
        "description": "Web Development",
        "quantity": 40,
        "unitPrice": 100,
        "taxRate": 20
      }
    ]
  }'
```

Get Dashboard Metrics

```
curl http://localhost:4000/api/v1/reports/dashboard \
  -H "Authorization: Bearer $TOKEN"
```

End of API Reference

Revision #3
Created 2026-02-23 10:47:52 UTC by John
Updated 2026-05-31 20:02:34 UTC by John