Backend — Target Architecture

Bilko API — Express Backend

BookStack — Provjeri PRVO

Prije traženja bilo čega — provjeri BookStack (http://localhost:6875). Centralna baza znanja za tools, skills, hooks, agents, rules, projekte, klijente, dokumentaciju. Ako odgovor postoji tamo — NE TRAŽI dalje.

Status: NOT BUILT YET

This directory is EMPTY. The CLAUDE.md describes the target architecture. When building, follow `docs/backend/API-REFERENCE.md` as the implementation contract.

Target Tech Stack

- **Framework:** Express + TypeScript
- **Database:** PostgreSQL 15 via Prisma
- **Auth:** JWT (15min access + 7d refresh) + Passport.js
- **Validation:** Zod
- **Middleware:** helmet, cors, rate-limit, auth-guard, zod-validation, error-handler

Route Structure

All routes under `/api/v1/{resource}`:

- `/api/v1/auth` — login, register, refresh, logout
- `/api/v1/organizations` — org CRUD
- `/api/v1/users` — user management
- `/api/v1/accounts` — chart of accounts
- `/api/v1/invoices` — invoice CRUD
- `/api/v1/expenses` — expense CRUD

- `/api/v1/transactions` — transaction ledger
- `/api/v1/contacts` — customer/vendor contacts
- `/api/v1/banking` — bank account integration
- `/api/v1/reports` — financial reports

Middleware Stack (Order Matters)

1. **helmet** — Security headers
2. **cors** — CORS with whitelist
3. **express.json()** — Body parser
4. **rate-limit** — 100 req/15min per IP
5. **auth-guard** — JWT validation (protected routes)
6. **zod-validation** — Request validation
7. **route-handler** — Business logic
8. **error-handler** — Centralized error responses

Error Response Format

```
{
  "error": "Error message",
  "code": "ERROR_CODE",
  "details": {} // optional
}
```

HTTP Status Codes:

- 400 — Validation error
- 401 — Unauthorized (missing/invalid token)
- 403 — Forbidden (insufficient permissions)
- 404 — Not found
- 500 — Internal server error

Database Access

- **ORM:** Prisma Client from `@bilko/database` package
- **Connection:** Read `DATABASE_URL` from env
- **Transactions:** Use Prisma transactions for multi-step operations
- **NEVER:** Raw SQL for business logic (use for migrations only)

Authentication

- **Strategy:** JWT (access + refresh tokens)
- **Access token:** 15min expiry, httpOnly cookie
- **Refresh token:** 7d expiry, httpOnly cookie, stored in DB
- **Password:** bcrypt hash with salt rounds = 12
- **2FA:** Optional TOTP (stored in User.twoFactorSecret)

Validation Rules

All requests validated with Zod schemas:

- **Money:** Must be string or number, converted to Decimal
- **Currency:** 3-letter ISO code (EUR, RSD, BAM, HRK)
- **Dates:** ISO 8601 format (YYYY-MM-DD)
- **UUIDs:** Valid v4 UUIDs for all IDs
- **Emails:** RFC 5322 compliant

Double-Entry Rules (CRITICAL)

Every financial transaction MUST:

1. Have both debit and credit accounts
2. Equal amounts (debit = credit)
3. Reference the source (invoice ID, expense ID)
4. Lock exchange rate at transaction date
5. Create audit log entry (LoggedAction)

Development Rules

1. **NEVER hold money** — This is an accounting tool, not a payment processor
2. **Immutable transactions** — Once locked, NEVER modify
3. **Audit everything** — All mutations logged to LoggedAction
4. **Multi-currency always** — Even single-currency orgs need exchange rate support
5. **Test with real accounting scenarios** — Invoice → payment → reconciliation

API Reference

Full endpoint documentation in `docs/backend/API-REFERENCE.md` (to be created). This file will be the contract for implementation.

Revision #4
Created 2026-02-23 10:24:30 UTC by John
Updated 2026-07-05 20:02:16 UTC by John