Environment Configuration

Bilko — Development Environment Setup

This guide walks through setting up a local development environment for Bilko.

Environment Configuration Overview

```
graph TD
    subgraph DEV["Development Environment"]
        D_ENV["apps/api/.env\napps/web/.env.local"]
        D_PG["PostgreSQL 15\nlocalhost:5432\nbilko_dev"]
        D_WEB["Next.js\nlocalhost:3000"]
        D_API["Express\nlocalhost:4000"]
        D_PRISMA["Prisma Studio\nlocalhost:5555"]
    end

    subgraph STAGING["Staging / Preview"]
        S_SECRETS["Railway Dashboard Env Vars\n(staging environment)"]
        S_VERCEL["Vercel Preview\nbilko-pr-{n}.vercel.app"]
        S_RAIL["Railway Staging\nbilko-api-staging"]
        S_PG["Railway PostgreSQL\nbilko_staging"]
    end

    subgraph PROD["Production"]
        P_SECRETS["Railway + Vercel\nDashboard Secrets"]
        P_WEB["Vercel Production\nbilko.io"]
        P_API["Railway Production\napi.bilko.io"]
        P_PG["Railway PostgreSQL\nbilko_prod"]
        P_R2["Cloudflare R2\nbilko-receipts"]
```

```
    end

    D_ENV --> D_API --> D_PG
    D_ENV --> D_WEB
    D_API --> D_PRISMA

    S_SECRETS --> S_RAIL --> S_PG
    S_SECRETS --> S_VERCEL

    P_SECRETS --> P_API --> P_PG
    P_SECRETS --> P_WEB
    P_API --> P_R2
```

Prerequisites

Required Software

Software	Version	Check Command	Install
Node.js	18+	`node --version`	https://nodejs.org
npm	9+	`npm --version`	Included with Node.js
PostgreSQL	15+	`psql --version`	https://postgresql.org/download
Git	Latest	`git --version`	https://git-scm.com

Optional Tools

Tool	Purpose	Install
Prisma Studio	Database GUI	`npx prisma studio`
Postman	API testing	https://postman.com
VS Code	Recommended IDE	https://code.visualstudio.com

Installation Steps

1. Clone Repository

```
git clone https://github.com/your-org/bilko.git
cd bilko
```

2. Install Dependencies

```
# Install all workspace dependencies
npm install
```

This installs dependencies for:

- Root workspace (Turborepo)
- `apps/web` (Next.js frontend)
- `apps/api` (Express backend)
- `packages/database` (Prisma)
- `packages/ui` (shared UI components)

Local Setup Flow

```
flowchart TD
    CLONE["git clone bilko"]
    INSTALL["npm install\n(Turborepo workspace)"]
    PG{{"PostgreSQL\navailable?"}}
    LOCAL_PG["Create local DB\npsql -U postgres\nCREATE DATABASE bilko_dev"]
    DOCKER_PG["Docker PostgreSQL\npostgres:15\nport 5432"]
    ENV["Configure .env files\napps/api/.env\napps/web/.env.local"]
    MIGRATE["npx prisma migrate dev\n(applies 15 table schema)"]
    GENERATE["npx prisma generate\n(Prisma Client)"]
    SEED["npx prisma db seed\n(demo org + user) — optional"]
    DEV["npm run dev\nlocalhost:3000 + 4000"]

    CLONE --> INSTALL --> PG
    PG -->|"Local install"| LOCAL_PG --> ENV
    PG -->|"Docker"| DOCKER_PG --> ENV
    ENV --> MIGRATE --> GENERATE --> SEED --> DEV
```

3. Set Up PostgreSQL Database

Option A: Local PostgreSQL Installation

Create database and user:

```
psql -U postgres
```

```
CREATE DATABASE bilko_dev;
CREATE USER bilko WITH PASSWORD 'bilko';
GRANT ALL PRIVILEGES ON DATABASE bilko_dev TO bilko;
\q
```

Option B: Docker PostgreSQL

```
docker run --name bilko-postgres \
  -e POSTGRES_USER=bilko \
  -e POSTGRES_PASSWORD=bilko \
  -e POSTGRES_DB=bilko_dev \
  -p 5432:5432 \
  -d postgres:15
```

4. Configure Environment Variables

apps/api/.env

Create `.env` file in `apps/api/` directory:

```
# Database
DATABASE_URL=postgresql://bilko:bilko@localhost:5432/bilko_dev

# JWT Secrets (use `openssl rand -base64 32` to generate)
JWT_SECRET=your-secret-here-change-in-production
JWT_REFRESH_SECRET=your-refresh-secret-here-change-in-production

# Email (optional for local dev, required for staging/production)
SENDGRID_API_KEY=

# File Storage (optional for local dev)
R2_ACCESS_KEY_ID=
R2_SECRET_ACCESS_KEY=
R2_BUCKET_NAME=bilko-receipts-dev
R2_ENDPOINT=

# App Config
PORT=4000
```

```
NODE_ENV=development
ALLOWED_ORIGINS=http://localhost:3000
```

apps/web/.env.local

Create `.env.local` file in `apps/web/` directory:

```
# API URL (backend)
NEXT_PUBLIC_API_URL=http://localhost:4000

# App Environment
NEXT_PUBLIC_APP_ENV=development
```

5. Run Database Migrations

```
cd packages/database
npx prisma migrate dev
npx prisma generate
```

This will:

1. Apply all migrations to `bilko_dev` database
2. Create 15 tables from `schema.prisma`
3. Generate Prisma Client

6. Seed Database (Optional)

Create seed script: `packages/database/prisma/seed.ts`

```
import { PrismaClient } from '@prisma/client';

const prisma = new PrismaClient();

async function main() {
  // Create demo organization
  const org = await prisma.organization.create({
    data: {
      name: 'Demo Company d.o.o.',
      registrationNumber: '12345678',
      vatNumber: 'RS123456789',
      baseCurrency: 'RSD',
```

```
      country: 'RS',
      language: 'sr',
    },
  });

  // Create demo user
  await prisma.user.create({
    data: {
      organizationId: org.id,
      email: 'demo@bilko.io',
      passwordHash: '$2b$12$...', // bcrypt hash of "demo123"
      fullName: 'Demo User',
      role: 'owner',
    },
  });

  console.log('✅ Seed data created');
}

main()
  .catch((e) => {
    console.error(e);
    process.exit(1);
  })
  .finally(async () => {
    await prisma.$disconnect();
  });
```

Run seed:

```
npx prisma db seed
```

Running the Application

Start All Services (Recommended)

From root directory:

```
npm run dev
```

Turborepo starts:

- **Frontend:** http://localhost:3000 (Next.js)
- **Backend:** http://localhost:4000 (Express)

Start Individual Services

Frontend Only

```
cd apps/web
npm run dev
```

Backend Only

```
cd apps/api
npm run dev
```

Development Tools

Prisma Studio (Database GUI)

```
cd packages/database
npx prisma studio
```

Opens at http://localhost:5555

Features:

- Browse all tables
- Edit records
- Run queries
- View relations

Hot Reload

Both frontend and backend support hot reload:

- **Frontend:** File changes trigger automatic browser refresh
- **Backend:** `nodemon` restarts server on `.ts` file changes

Tech Stack Overview

Frontend (apps/web/)

Technology	Version	Purpose
Next.js	15.0.0	React framework with SSR
React	19.0.0	UI library
TypeScript	5.3.0	Type safety
Tailwind CSS	4.0.0	Styling
shadcn/ui	Latest	Component library (Radix UI + Tailwind)
Zustand	4.5.0	State management
Recharts	2.15.0	Charts (revenue, expenses)
Lucide React	Latest	Icons

Backend (apps/api/)

Technology	Version	Purpose
Express	TBD	Web framework
TypeScript	5.3.0	Type safety
Prisma	Latest	ORM + migrations
PostgreSQL	15+	Database
Passport.js	TBD	Authentication
Zod	TBD	Validation
Helmet	TBD	Security headers
bcrypt	TBD	Password hashing
jsonwebtoken	TBD	JWT tokens

Database (packages/database/)

Feature	Implementation
ORM	Prisma
Database	PostgreSQL 15
Models	15 (see schema.prisma)
Migrations	Prisma Migrate
Seeding	prisma/seed.ts

Common Tasks

Create Database Migration

After modifying `schema.prisma`:

```
cd packages/database
npx prisma migrate dev --name describe_your_changes
```

Reset Database (DEV ONLY)

WARNING: Deletes all data.

```
cd packages/database
npx prisma migrate reset
```

Generate Prisma Client

After pulling new migrations:

```
cd packages/database
npx prisma generate
```

Run Linter

```
npm run lint
```

Runs ESLint + Prettier on all workspaces.

Run Type Check

```
npm run type-check
```

Runs TypeScript compiler in `--noEmit` mode (checks types without building).

Build for Production

```
npm run build
```

Builds:

- `apps/web/.next/` — Next.js production build
- `apps/api/dist/` — Compiled TypeScript

Troubleshooting

Database Connection Errors

Error: `Can't reach database server at localhost:5432`

Solutions:

1. Check PostgreSQL is running: `pg_isready`
2. Verify credentials in `.env`
3. Check port 5432 is not blocked

Port Already in Use

Error: `Port 3000 is already in use`

Solutions:

1. Kill process using port: `lsof -ti:3000 | xargs kill`
2. Change port: `PORT=3001 npm run dev`

Prisma Client Not Generated

Error: `@prisma/client` not found

Solution:

```
cd packages/database
npx prisma generate
```

TypeScript Errors After Pulling Changes

Solution:

```
npm install
npx prisma generate
npm run type-check
```

Hot Reload Not Working

Solution:

1. Restart dev server
2. Clear Next.js cache: `rm -rf apps/web/.next`
3. Check file watcher limits (Linux): `sysctl fs.inotify.max_user_watches`

VS Code Configuration

Recommended Extensions

Create `.vscode/extensions.json`:

```
{
  "recommendations": [
    "dbaeumer.vscode-eslint",
    "esbenp.prettier-vscode",
    "bradlc.vscode-tailwindcss",
    "prisma.prisma",
    "ms-vscode.vscode-typescript-next"
```

```
  ]
}
```

Settings

Create `.vscode/settings.json`:

```
{
  "editor.formatOnSave": true,
  "editor.defaultFormatter": "esbenp.prettier-vscode",
  "editor.codeActionsOnSave": {
    "source.fixAll.eslint": true
  },
  "typescript.tsdk": "node_modules/typescript/lib",
  "tailwindCSS.experimental.classRegex": [
    ["cva\\(([^)]*)\\)", "[\"'`]([^\"'`]*).*?[\"'`]"]
  ]
}
```

Secrets Management

```
flowchart LR
    DEV_SECRET["Developer\n.env file\n(gitignored)"]
    GH_SECRET["GitHub Secrets\nActions →
Settings\nVERCEL_TOKEN\nRAILWAY_TOKEN\nTEST_DATABASE_URL\nSLACK_WEBHOOK"]
    VERCEL_ENV["Vercel Dashboard\nEnvironment
Variables\nNEXT_PUBLIC_API_URL\nNEXT_PUBLIC_APP_ENV"]
    RAILWAY_ENV["Railway Dashboard\nEnvironment Variables\nDATABASE_URL
(auto)\nJWT_SECRET\nJWT_REFRESH_SECRET\nSENDGRID_API_KEY\nR2_ACCESS_KEY_ID\nR2_SECRET_ACCESS_K
EY"]

    subgraph NEVERCOMMIT["NEVER commit to git"]
        SECRET_FILE[".env files\nAPI keys\nJWT secrets\nDB passwords"]
    end

    DEV_SECRET -->|"local only"| NEVERCOMMIT
    GH_SECRET -->|"CI/CD pipeline"| VERCEL_ENV
```

```
    GH_SECRET -->|"CI/CD pipeline"| RAILWAY_ENV
```

Environment Variables Reference

apps/api/.env

Variable	Required	Default	Description
`DATABASE_URL`	Yes	—	PostgreSQL connection string
`JWT_SECRET`	Yes	—	Access token secret (32+ chars)
`JWT_REFRESH_SECRET`	Yes	—	Refresh token secret (32+ chars)
`SENDGRID_API_KEY`	No	—	SendGrid API key (emails)
`R2_ACCESS_KEY_ID`	No	—	Cloudflare R2 access key
`R2_SECRET_ACCESS_KEY`	No	—	Cloudflare R2 secret key
`R2_BUCKET_NAME`	No	—	R2 bucket name
`R2_ENDPOINT`	No	—	R2 endpoint URL
`PORT`	No	4000	API server port
`NODE_ENV`	No	development	Environment (development/production)
`ALLOWED_ORIGINS`	No	*	CORS allowed origins (comma-separated)

apps/web/.env.local

Variable	Required	Default	Description
`NEXT_PUBLIC_API_URL`	Yes	—	Backend API URL
`NEXT_PUBLIC_APP_ENV`	No	development	Environment name

Testing Locally

Unit Tests

```
npm run test:unit
```

Integration Tests

Requires test database:

```
# Create test database
createdb bilko_test

# Run tests
npm run test:integration
```

E2E Tests

Requires both frontend and backend running:

```
# Terminal 1: Start dev servers
npm run dev

# Terminal 2: Run E2E tests
npm run test:e2e
```

Next Steps

After setting up your environment:

1. **Read the docs:**
   - Backend API Reference
   - Frontend Component Guide
   - Database Schema
2. **Create your first feature:**
   - Pick a task from the backlog
   - Create feature branch: `git checkout -b feature/your-feature`
   - Make changes, test locally
   - Submit PR
3. **Join the team:**
   - Slack: #bilko-dev
   - Weekly sync: Fridays 10:00 CET

- Documentation: Bilko Wiki

Related Documents

- Deployment Guide: DEPLOYMENT.md
- CI/CD Pipeline: CI-CD.md
- Security Architecture: ../security/SECURITY-ARCHITECTURE.md

---

Last Updated: 2026-02-20 **Status:** CURRENT — Reflects actual setup as of this date **Maintainer:** John (AI Director)

Revision #5
Created 2026-02-23 10:48:10 UTC by John
Updated 2026-05-31 20:02:47 UTC by John