Project Elevate

Codebase Architecture Overview

---

1. Technology Stack and Core Dependencies

- **Backend Framework:** Fastify (high-performance web framework)
- **Language:** TypeScript
- **Database:** PostgreSQL with Drizzle ORM
- **Authentication:** JWT with cookie-based sessions
- **Validation:** Zod schemas for type-safe validation
- **Testing:** Jest with coverage reporting
- **Frontend:** HTMX for dynamic UI interactions

---

2. Project Structure

```
src/
├── app.ts              # Main application entry point
├── applications/       # Business application management
├── auth/               # JWT authentication system
├── billing/            # Billing and documents
├── company/            # Company management
├── db/
│   └── schema/         # Database schema definitions
├── owners/             # Company ownership tracking
├── templates/          # HTML template components
├── types/              # TypeScript type definitions
└── user/               # User management
```

---

3. Database Schema and Core Entities

Users (src/db/schema/users.ts:9)

- Handles user authentication and Role-Based Access Control (RBAC).
- Supported roles: **admin**, **manager**, **user**.
- Users are linked to companies via a **companyId** foreign key.

Companies (src/db/schema/companies.ts:8)

- Responsible for company registration and management.
- Supports hierarchical structures with **parentCompanyId**.
- Related entities: financials, documents, users.

Applications (src/db/schema/applications.ts:20)

- Manages business applications (e.g., **FUNDING**, **BUSINESS_RESCUE**).
- Tracks application status through the following workflow:
  STARTED → PENDING_DOCS → PENDING_PAYMENT → APPROVED/REJECTED
- Includes a **JSON properties field** for flexible data storage.

---

4. Authentication System (src/auth/jwt.auth.plugin.ts:13)

- JWT-based authentication with signed cookies.
- Cookie name: **access_token**.
- Role-based authorization middleware for fine-grained access control.
- Automatic token verification for all protected routes.

---

5. API Architecture

The system follows a **hybrid REST + HTMX pattern**:

Dual Response System (src/user/users.routes.ts:151):

- Routes provide both **JSON (API)** and **HTML (HTMX)** responses.
- Content negotiation is based on Accept headers and hx-request.
- Example: GET /users
  - Returns JSON for API calls.
  - Returns an HTML table for browser/HTMX requests.

HTMX Integration:

- Server-side rendered components are stored in **src/templates/**.
- Supports real-time UI updates without full page reloads.
- Enables out-of-band swaps for complex interactions.

---

6. Key Features and Business Logic

Role-Based Access Control (RBAC):

- **Admin:** Full system access, including user management.
- **Manager:** Company-scoped access to users and data.
- **User:** Basic access to own company data.

Application Workflow (src/applications/applications.routes.ts):

- Multi-step application process with clear status tracking.
- Document upload and management integrated into workflow.
- Company-specific application filtering.

Company Management:

- Hierarchical company structure with parent/child relationships.
- Financial data tracking across defined time periods.
- Document storage with S3 URLs for secure access.

---

7. Development Environment

- **Start command:** npm run dev (uses Nodemon + TSX).
- **Database:** Docker Compose setup for PostgreSQL.
- **Migrations:** Managed with Drizzle Kit.
- **Testing:** Jest with HTML reporting for test coverage.

---

8. Security Patterns

- Password hashing with **bcrypt** (10 salt rounds).
- JWT tokens stored in **httpOnly cookies**.

- Input validation with **Zod schemas**.
- SQL injection protection using **Drizzle ORM**.
- Role-based access control enforced at the route level.

---

9. UI Architecture

The application provides multiple dashboard types:

- **Main Dashboard:** Landing page (src/app.ts:141).
- **User/Manager Dashboard:** Role-specific views (src/app.ts:147).
- **Admin Dashboard:** Full system oversight (src/app.ts:156).

Each module (users, companies, applications, etc.) includes dedicated UI components for:

- List views.
- Detailed forms.
- Dynamic interactions powered by **HTMX**.

Conclusion

Project Elevate is a well-structured business management platform that combines:

- Clear separation of concerns across modules.
- A robust authentication and security model.
- Modern full-stack architecture blending **server-side rendering** with **dynamic client interactions**.

This architecture ensures scalability, maintainability, and a strong foundation for future development.