CLAUDE.md

This file provides guidance to Claude Code (claude.ai/code) when working with code in this repository.

Project Overview

Sauberfix is a cleaning service appointment management API built with .NET 9 and Visual Basic, using PostgreSQL as the database and Entity Framework Core for data access. The application provides JWT-based authentication for managing customers (Kunden), employees (Mitarbeiter), and appointments (Termine).

Technology Stack

- **Framework**: .NET 9.0
- **Languages**: Visual Basic (.vb) for main application, C# for data layer
- **Database**: PostgreSQL with EF Core 9.0
- **Authentication**: JWT Bearer tokens
- **Web Framework**: ASP.NET Core Minimal APIs

Common Commands

Running the Application

```
dotnet run
```

The app automatically runs migrations and seeds the database on startup. Runs on port 5000 by default.

Building

```
dotnet build
```

Database Migrations

Create a new migration (from Sauberfix.Data directory):

```
cd Sauberfix.Data
dotnet ef migrations add MigrationName --startup-project ../sauberfix.vbproj
```

Apply migrations (automatically happens on `dotnet run`, but manually):

```
dotnet ef database update --startup-project ../sauberfix.vbproj
```

Testing Database Connection

The app will fail on startup if PostgreSQL connection is invalid. Check `appsettings.json` connection string.

Project Structure

Root Project (`sauberfix.vbproj`)

- **Program.vb**: Application entry point, configures services, JWT authentication, and defines all API endpoints using Minimal APIs pattern
- **Dtos.vb**: All DTOs (Data Transfer Objects) for requests and responses
- **DatabaseSeeder.vb**: Seeds initial admin user on first run
- **Services/**: Business logic layer
  - `AuthService.vb`: Handles login and JWT token generation using SHA256 password hashing
  - `KundenService.vb`: Customer CRUD operations with 3NF Ort (location) normalization
  - `MitarbeiterService.vb`: Employee CRUD operations
  - `TerminService.vb`: Appointment CRUD with 60-minute duration and collision detection

  - `ErinnerungsService.vb`: Background service that checks every minute for appointments 24 hours away and logs reminder emails (not actually sent)

Data Project (`Sauberfix.Data.csproj`)

C# class library containing: - **AppDbContext.cs**: EF Core DbContext - **Entity Models** (all in C#): - `Kunde.cs`: Customer entity with normalized Ort relationship - `Mitarbeiter.cs`: Employee entity with role-based access (Admin/User) - `Termin.cs`: Appointment entity with status enum (Geplant/Erledigt/Storniert) and reminder tracking - `Ort.cs`: Location entity (PLZ/Stadt) for 3rd normal form compliance - **Migrations/**: EF Core migration files

Architecture Notes

Authentication & Authorization

- JWT tokens issued on successful login with 8-hour expiration
- Tokens include claims: NameIdentifier (user ID), Name (username), Role (Admin/User)
- Admin users see all appointments; regular users see only their own
- Password hashing uses SHA256 (stored in `Mitarbeiter.PasswordHash`)
- Most endpoints require authorization except `/login`, `POST /kunden`, and `GET /kunden`

Data Model Relationships

- **Kunde** → **Ort**: Many-to-one (3NF normalization)
- **Termin** → **Kunde**: Many-to-one
- **Termin** → **Mitarbeiter**: Many-to-one (nullable)
- Appointments are 60 minutes long (hardcoded in TerminService)
- Collision detection prevents overlapping appointments per employee

Service Layer Pattern

All business logic resides in service classes injected via DI: - Services are scoped (one instance per request) - ErinnerungsService is a hosted background service - All services accept `AppDbContext` via constructor injection - Services handle validation, throw `ArgumentException` on errors - API endpoints in Program.vb call service methods and map exceptions to HTTP error responses

Background Service

`ErinnerungsService` runs continuously: - Checks every 1 minute for appointments occurring in 24 hours (±2 minutes) - Only processes appointments with `Status = Geplant` and `ErinnerungVerschickt = false` - Logs email reminders (actual email sending not implemented) - Updates `ErinnerungVerschickt` flag to prevent duplicate notifications

Configuration Management

- `appsettings.json` is gitignored (contains secrets)
- `appsettings.Template.json` provides the template structure
- Required settings:
  - `ConnectionStrings:DefaultConnection`: PostgreSQL connection
  - `JwtSettings:Key`: Must be at least 32 characters
  - `JwtSettings:Issuer` and `Audience`: Set to "SauberfixAPI" and "SauberfixClient"

Important Implementation Details

- `AppContext.SetSwitch("Npgsql.EnableLegacyTimestampBehavior", True)` in Program.vb handles timestamp compatibility

- Database migrations are stored in `Sauberfix.Data` but require `--startup-project ../sauberfix.vbproj` flag
- Frontend is a single static HTML file in `wwwroot/index.html`
- Default admin credentials after first run: username `admin`, password `admin123`

Language Mixing

The codebase intentionally uses: - **Visual Basic** for application logic (Program.vb, Services/, DTOs, DatabaseSeeder) - **C#** for data models and EF Core migrations (Sauberfix.Data project)

When adding new features: - Follow existing language conventions per directory - New services → Visual Basic in Services/ - New entities → C# in Sauberfix.Data/ - API endpoints → Add to Program.vb using Minimal API MapGet/MapPost/MapPut/MapDelete