



TECHNICAL GUIDE

Lead Generation System

ASP.NET Core (.NET 10) Web API + Entity Framework Core + SQL Server, built with Clean Architecture and SOLID principles. Includes a built-in web UI and Swagger.

.NET 10

C#

Entity Framework Core

SQL Server

Clean Architecture

SOLID

VERSION

1.0.0

DATABASE

LGSDb

DATE

October 2026

Contents

01	Run	Start the application locally
02	Installer	Setup wizard, service, database
03	Sign in	Default account and authentication
04	Solution layout	Projects and dependencies
05	How SOLID is applied	Principles in the code
06	Features	What the system does
07	API	Endpoint reference
08	Migrations	Database schema changes

1 Run

POWERSHELL

```
cd LeadGenSystem
dotnet run --project src/LeadGen.Api --launch-profile http
```

- Web UI: <http://localhost:5019>
- Swagger: <http://localhost:5019/swagger>

Pending migrations are applied on startup (`Database:MigrateOnStartup` in `appsettings.json`), so `LGSDb` is created automatically if it does not exist.

DEVELOPMENT SETTINGS

For local development, the connection string and default admin password live in `src/LeadGen.Api/appsettings.Development.json` , which is never included in the installer. `appsettings.json` holds no credentials. `TrustServerCertificate=True` is required because the SQL Server client encrypts connections by default and a local SQL Server uses a self-signed certificate.

2 Installer

`installer\output\LeadGenerationSystem-Setup-1.0.0.exe` installs the system on any 64-bit Windows 10/11 or Windows Server PC. No .NET runtime is needed (the app is self-contained), but SQL Server must already be installed and reachable.

The setup wizard asks for

- 1 Install folder** (default `C:\Program Files\Lead Generation System`).
- 2 Database:** SQL Server instance, database name (default `LGSDb`), and SQL user/password. Leave the user empty for Windows authentication. The connection is tested before you continue.
- 3 Administrator account:** username and password (8+ characters) for the first sign-in. Ignored when upgrading an existing database.
- 4 Web server:** port (default `5019`) and, optionally, access from other computers (adds a Windows Firewall rule).

It then

- creates or upgrades the database and creates the admin user;

- installs and starts the **Lead Generation System** Windows service, which starts automatically at boot and restarts if it fails;
- adds Start-menu and (optionally) desktop shortcuts that open `http://localhost:<port>`.

SETTINGS & UPGRADES

Settings are written to `appsettings.Production.json` in the install folder. Only Administrators and SYSTEM can read that file, and the admin password is removed from it once the database is ready. Running the installer again upgrades in place and remembers your previous answers (not passwords). Uninstalling removes the service, shortcuts and firewall rule but **keeps the database**.

The `Database` folder in the install directory contains `LGSDb.sql` (creates the database and all tables; safe to run more than once) for DBAs who prefer to create it manually.

Building the installer

```
winget install JRSoftware.InnoSetup          # once
.\installer\Build-Installer.ps1 -Version 1.0.0
```

POWERSHELL

This publishes the app, generates the database scripts, creates the icon from `wwwroot\img\logo.png` and compiles `installer/LeadGenSystem.iss`.

3 Sign in

The app opens on a login page. On first start, if the `Users` table is empty, an admin account is created from `Auth:DefaultAdmin` (in development: `appsettings.Development.json`; when installed: the account you enter in the setup wizard):

U Username

`admin` (or `admin@leadgen.local`)

P Password

`Admin@123`

CHANGE BEFORE DEPLOYING

The default password is for development only. In Development (`Auth:ShowDefaultCredentials` in `appsettings.Development.json`) the login page shows these credentials with a **Use** button to fill them in. The setting is off by default, so other environments never expose them.

Authentication uses an `HttpOnly, SameSite=Strict` cookie (8-hour sliding session; "Keep me signed in" makes it persistent). Every API endpoint requires sign-in except `POST /api/auth/login`, which is limited to 10 attempts per minute per IP. Passwords are stored as PBKDF2-SHA256 hashes.

METHOD	ROUTE
POST	<code>/api/auth/login</code> — <code>{ "login": "admin", "password": "...", "rememberMe": false }</code>
POST	<code>/api/auth/logout</code>
GET	<code>/api/auth/me</code>

4 Solution layout

LeadGen.Api

Controllers, global exception handling, DI composition root, web UI.

LeadGen.Infrastructure

EF Core `DbContext`, Fluent configurations, migrations, repositories, unit of work. Implements Application abstractions.

LeadGen.Application

Use cases: service interfaces + implementations, DTOs, repository abstractions, lead-scoring rules. Depends only on Domain.

LeadGen.Domain

Entities, enums, domain behaviour. No dependencies.

Dependency direction: `Api` → `Infrastructure` → `Application` → `Domain`

5 How SOLID is applied

S Single Responsibility

One service per use-case area (`LeadService` , `LeadActivityService` , `CampaignService` , `LeadSourceService` , `DashboardService`); reporting queries live in `LeadStatistics` , not in the CRUD repository; exception→HTTP mapping lives only in `GlobalExceptionHandler` .

O Open/Closed

Lead scoring: `LeadScoringService` sums every registered `ILeadScoringRule` . Add a rule class and register it in `Application/DependencyInjection.cs` — no existing code changes.

L Liskov Substitution

`LeadRepository` , `LeadSourceRepository` , `LeadActivityRepository` extend `Repository<T>` and can be used anywhere `IRepository<T>` is expected.

I Interface Segregation

`IReadRepository<T>` vs `IRepository<T>` ; `ILeadStatistics` is separate from `ILeadRepository` ; `IUnitOfWork` only saves.

D Dependency Inversion

Application services depend on interfaces (`ILeadRepository` , `IUnitOfWork` , ...); Infrastructure provides implementations, wired in the API's composition root.

6 Features

- **Leads** – CRUD, search, filter by status/source/campaign/min score, sort, paging; unique email.
- **Pipeline** – every status change is logged as an activity.

New → Contacted → Qualified → ProposalSent → Negotiation → Won / Lost

- **Lead scoring (0–100)** – recalculated on create/update or via `POST /api/leads/{id}/rescore` .

30

Profile completeness

20

Business email domain

25

Job-title seniority

25

Estimated deal value

- **Activities** – notes, calls, emails, meetings per lead.
- **Campaigns** and **Lead sources** – CRUD with lead counts; a source in use can't be deleted (deactivate it instead). Six sources are seeded.
- **Dashboard** – totals, new this month, conversion rate (won / closed), open pipeline value, breakdowns by status and source.
- Errors are returned as RFC 7807 problem details: **400** validation, **404** not found, **409** duplicate, **422** business rule.

7 API

METHOD	ROUTE
GET	/api/leads?search=&status=&leadSourceId=&campaignId=&minScore=&sortBy=createdAt score name value&sortDesc=true&page=1&pageSize=20
GET PUT DELETE	/api/leads/{id}
POST	/api/leads
PATCH	/api/leads/{id}/status — { "status": "Qualified", "reason": "..." }
POST	/api/leads/{id}/rescore
GET POST	/api/leads/{id}/activities
DELETE	/api/leads/{id}/activities/{activityId}
GET POST GET PUT DELETE {id}	/api/campaigns , /api/lead-sources {id}
GET	/api/dashboard

8 Migrations

POWERSHELL

```
# Add a migration
dotnet ef migrations add <Name> `
  --project src/LeadGen.Infrastructure `
  --startup-project src/LeadGen.Api `
  --output-dir Persistence/Migrations `
  -- --environment Development

# Apply migrations to the database
dotnet ef database update `
  --project src/LeadGen.Infrastructure `
  --startup-project src/LeadGen.Api `
  -- --environment Development
```