English version

Table of Contents

AIWeek Search Engine

Fullstack platform for advanced contact management and search, featuring AI-powered natural language queries. Includes a .NET Core backend and a React frontend, JWT authentication, roles, advanced filters, CSV export, and a modern robust UX.

Official repository: https://github.com/TINAlbert/AIWeek-SearchEngine

---

Main features

- Simple and advanced contact search (name, email, company, profiles, etc.)
- **AI Search**: natural language queries, SQL generation and execution via local LLM (Ollama)
- Export contacts to CSV
- User, role, and avatar management
- Modern, responsive, accessible UI
- JWT security and refresh token
- Reusable advanced filter history

Architecture

- `/backend/` — .NET Core 7+ REST API, Entity Framework Core, Identity, JWT
- `/frontend/` — React 19+, Tailwind CSS, Axios, Context API

See detailed docs:

- Backend README

- Frontend README

Technical requirements

- **Backend:** .NET Core 7+, Entity Framework Core, Microsoft Identity, JWT, AutoMapper, Scalar (OpenAPI), xUnit
- **Frontend:** React 19+, Vite, Tailwind CSS, Axios, React Context, React Router DOM, React Hook Form, Yup
- **AI:** Ollama (`llama3` or `sqlcoder` model)

Installation & Getting Started

1. Backend

```
cd backend/SearchServiceEngine
# Restore packages and apply migrations
dotnet restore
dotnet ef database update
# Run backend
dotnet run
```

Default API: `http://localhost:5252/api`

> **Note:** The initial database seed for testing creates two users: `Admin` (password: `Admin123!`) and `User` (password: `User123!`). These are for testing purposes only and should not be used in production.

2. Frontend

```
cd frontend
npm install
# Set API URL in .env
# VITE_API_BASE_URL=http://localhost:5252/api
npm run dev
```

Default app: `http://localhost:5173`

3. Ollama (AI) setup

1. Install Ollama: https://ollama.com/download
2. Download the LLM model:

```
ollama pull llama3
# or
ollama pull sqlcoder
```

3. Start the service:

```
ollama serve
ollama run llama3
```

   The backend expects Ollama at `http://localhost:11434`.
4. Check connectivity with `/api/ai/ping`.

Using AI Search

- Access from the sidebar: **AI Search** ("Sparkles" icon)
- Describe your query in natural language (e.g., "Active contacts in Madrid")
- The AI generates and executes the SQL, showing results and the generated query
- Only authenticated users can access
- The backend only executes SQL generated that is a SELECT
- Extended timeout for AI requests (up to 5 minutes)

How it works:

- The frontend sends a natural language prompt to the backend
- The backend attaches the real database schema (in SQL CREATE TABLE format) to the prompt
- Ollama (with a model like `llama3` or `sqlcoder`) generates a safe SQL SELECT query
- The backend validates and executes only SELECT queries, returning results and the generated SQL
- Results and the generated SQL are shown in a modern, responsive table
- Only authenticated users can access this feature
- Extended timeout for AI requests (up to 5 minutes)

Ollama setup:

1. Download and install Ollama: https://ollama.com/download
2. Download a suitable model (recommended: `llama3` or `sqlcoder`):

```
ollama pull llama3
# or
ollama pull sqlcoder
```

3. Start the Ollama service:

```
ollama serve
# (optional) ollama run llama3
```

   The backend expects Ollama at `http://localhost:11434`.
4. Check backend connectivity with `/api/ai/ping` endpoint.

Security:

- Only authenticated users can access AI endpoints
- The backend strictly validates that only SELECT queries are executed
- The database schema is sent to the LLM to ensure accurate and safe SQL generation

For more details, see the backend README.

Environment & config

- **Frontend:** `.env`:

```
VITE_API_BASE_URL=http://localhost:5252/api
```

- **Backend:** `appsettings.json`:

```
{
  "Jwt": {
    "Key": "SuperSecretKey12345678901234567890123456789012",
    "Issuer": "AIWeekIssuer",
    "Audience": "AIWeekAudience"
  },
  "ConnectionStrings": {
    "DefaultConnection": "Data Source=aiweek.db"
  },
  "AvatarsPath": "wwwroot/avatars",
  "SeedInitialData": true
}
```

Security

- JWT authentication (Bearer Token)
- Secure, revocable refresh token
- Roles and claims in backend and frontend
- Private route protection and role validation in UI
- Only authenticated users can access AI endpoints

Testing

- Backend: unit tests with xUnit and Moq
- Frontend: ready for React Testing Library

Documentation

- Backend README
- Frontend README

Credits

Developed by Albert G.M. (GitHub: TINAlbert). Based on best practices in architecture, security, and UX for modern web applications. Repository: https://github.com/TINAlbert/AIWeek-SearchEngine

---

Versión en castellano

Índice

AIWeek Search Engine

Plataforma fullstack para la gestión y búsqueda avanzada de contactos personales, con integración de IA para consultas en lenguaje natural. Incluye backend en .NET Core y frontend en React, autenticación JWT, roles, filtros avanzados, exportación CSV y una experiencia moderna y robusta.

Repositorio oficial: https://github.com/TINAlbert/AIWeek-SearchEngine

---

Características principales

- Búsqueda simple y avanzada de contactos (nombre, email, empresa, perfiles, etc.)
- **Búsqueda IA**: consulta en lenguaje natural, generación y ejecución de SQL vía LLM local (Ollama)
- Exportación de contactos a CSV
- Gestión de usuarios, roles y avatares
- UI moderna, responsiva y accesible
- Seguridad JWT y refresh token
- Historial reutilizable de filtros avanzados

Arquitectura y estructura

- `/backend/` — API REST en .NET Core 7+, Entity Framework Core, Identity, JWT
- `/frontend/` — React 19+, Tailwind CSS, Axios, Context API

Ver documentación detallada en:

- Documentación Backend (SearchServiceEngine)
- Documentación Frontend

Requisitos técnicos

- **Backend:** .NET Core 7+, Entity Framework Core, Microsoft Identity, JWT, AutoMapper, Scalar (OpenAPI), xUnit
- **Frontend:** React 19+, Vite, Tailwind CSS, Axios, React Context, React Router DOM, React Hook Form, Yup
- **IA:** Ollama (modelo `llama3` o `sqlcoder`)

Instalación y puesta en marcha

1. Backend

```
cd backend/SearchServiceEngine
# Restaurar paquetes y aplicar migraciones
 dotnet restore
 dotnet ef database update
# Ejecutar backend
 dotnet run
```

Por defecto, la API estará en `http://localhost:5252/api`.

> **Nota:** El seed inicial de la base de datos para testing crea dos usuarios: `Admin` (contraseña: `Admin123!`) y `User` (contraseña: `User123!`). Son solo válidos para pruebas y no deben usarse en producción.

2. Frontend

```
cd frontend
npm install
# Configura la URL de la API en .env
# VITE_API_BASE_URL=http://localhost:5252/api
npm run dev
```

La app estará en `http://localhost:5173`.

3. Configuración y uso de Ollama (IA)

1. Instala Ollama: https://ollama.com/download
2. Descarga el modelo LLM:

   ```
   ollama pull llama3
   # o
   ollama pull sqlcoder
   ```

3. Inicia el servicio:

   ```
   ollama serve
   ollama run llama3
   ```

   El backend espera Ollama en `http://localhost:11434`.
4. Comprueba la conectividad con `/api/ai/ping`.

Uso de la Búsqueda IA

- Accede desde el menú lateral: **Búsqueda IA** (icono "Sparkles")
- Describe la consulta en lenguaje natural (ej: "Contactos activos de Madrid")
- La IA genera y ejecuta la SQL, mostrando resultados y la consulta generada
- Solo usuarios autenticados pueden acceder
- El backend solo ejecuta SQL generada que sea un SELECT
- Timeout extendido para peticiones de IA (hasta 5 minutos)

Variables de entorno y configuración

- **Frontend:** `.env` con:

```
VITE_API_BASE_URL=http://localhost:5252/api
```

- **Backend:** `appsettings.json` con:

```
{
  "Jwt": {
    "Key": "SuperSecretKey12345678901234567890123456789012",
    "Issuer": "AIWeekIssuer",
    "Audience": "AIWeekAudience"
  },
  "ConnectionStrings": {
    "DefaultConnection": "Data Source=aiweek.db"
  },
  "AvatarsPath": "wwwroot/avatars",
  "SeedInitialData": true
}
```

Seguridad

- Autenticación JWT (Bearer Token)
- Refresh token seguro y revocable
- Roles y claims en backend y frontend
- Protección de rutas privadas y validación de roles en la UI
- Solo usuarios autenticados pueden acceder a la IA

Testing

- Backend: pruebas unitarias con xUnit y Moq
- Frontend: preparado para React Testing Library

Documentación específica

- Documentación Backend (SearchServiceEngine)
- Documentación Frontend

Créditos y agradecimientos

Desarrollado por el programador Albert G.M. (GitHub: TINAlbert). Basado en mejores prácticas de arquitectura, seguridad y experiencia de usuario para aplicaciones web modernas. Repositorio: https://github.com/TINAlbert/AIWeek-SearchEngine