paysky-task Backend Project Structure & Explanation

Overview

This is an ASP.NET Core Web API project (.NET 8) using Entity Framework Core for data access and SQL Server as the database. It implements a job application system with JWT authentication, role-based authorization, and background services.

Frameworks & Libraries Used

- **ASP.NET Core 8**: For building RESTful APIs.
- **Entity Framework Core 8**: For ORM/database access.
- **Microsoft SQL Server**: As the database.
- **JWT (System.IdentityModel.Tokens.Jwt)**: For authentication.
- **Swagger (Swashbuckle)**: For API documentation/testing.
- **Microsoft.Extensions.Hosting**: For background services.
- **Microsoft.Extensions.Caching.Memory**: For in-memory caching.

File-by-File Explanation

1. `Program.cs`

- **Purpose**: Entry point and configuration for the web API.
- **Responsibilities**:
  - Registers controllers, Swagger, DbContext, JWT authentication, memory cache, and background services.
  - Configures the HTTP request pipeline (middleware).
  - Starts the web server.

---

2. `appsettings.json` / `appsettings.Development.json`

- **Purpose**: Configuration files.
- **Responsibilities**:
  - Stores connection strings for SQL Server.
  - Stores JWT settings (key, issuer, audience).
  - Logging configuration.

---

3. `Data/ApplicationDbContext.cs`

- **Purpose**: Entity Framework Core database context.
- **Responsibilities**:
  - Defines `DbSet<T>` for `User`, `Vacancy`, and `Application` entities.
  - Configures relationships and cascade behaviors to avoid cycles (important for SQL Server).
  - Used for all database operations.

---

4. `Entities/Entities.cs`

- **Purpose**: Domain models/entities.
- **Responsibilities**:
  - `User`: Represents a user (Employer or Applicant).
  - `Vacancy`: Represents a job vacancy.
  - `Application`: Represents a job application by an applicant.
  - `UserRole`: Enum for user roles.

---

5. `DTOs/DTOs.cs`

- **Purpose**: Data Transfer Objects (DTOs) for API requests and responses.
- **Responsibilities**:
  - `RegisterDto`, `LoginDto`: For authentication endpoints.
  - `VacancyDto`: For creating/updating vacancies.
  - `VacancyViewDto`: For safely returning vacancy data (prevents object cycles).
  - `ApplicationDto`: For applying to vacancies.

---

6. `Controllers/AuthController.cs`

- **Purpose**: Handles authentication (register/login).
- **Responsibilities**:
  - `POST /api/Auth/register`: Register a new user.
  - `POST /api/Auth/login`: Login and receive a JWT token.
  - Hashes passwords using SHA256.

---

7. `Controllers/VacancyController.cs`

- **Purpose**: Handles employer actions on vacancies.
- **Responsibilities**:
  - Protected by `[Authorize(Roles = "Employer")]`.
  - `POST /api/vacancy`: Create a vacancy.
  - `PUT /api/vacancy/{id}`: Update a vacancy.
  - `DELETE /api/vacancy/{id}`: Delete a vacancy.
  - `POST /api/vacancy/{id}/deactivate`: Deactivate a vacancy.
  - `GET /api/vacancy`: List employer's vacancies (returns DTOs).
  - `GET /api/vacancy/{id}/applicants`: List applicants for a vacancy.

---

8. `Controllers/ApplicationController.cs`

- **Purpose**: Handles applicant actions.
- **Responsibilities**:
  - Protected by `[Authorize(Roles = "Applicant")]` (except search).

  - `GET /api/application/search`: Public search for active, non-archived, non-expired vacancies (returns DTOs).
  - `POST /api/application/apply`: Apply for a vacancy (enforces business rules: one application per 24h, max applications, etc.).

---

9. `Services/JwtTokenService.cs`

- **Purpose**: JWT token generation.
- **Responsibilities**:
  - Generates JWT tokens with user claims (id, username, role).
  - Uses settings from `appsettings.json`.

---

10. `Services/VacancyArchiverService.cs`

- **Purpose**: Background service for archiving expired vacancies.
- **Responsibilities**:
  - Runs every hour.
  - Finds vacancies where `ExpiryDate` is in the past and `IsArchived` is false.
  - Sets `IsArchived = true` and `IsActive = false`.
  - Logs archiving actions.

---

11. `Migrations/`

- **Purpose**: Entity Framework Core migration files.
- **Responsibilities**:
  - Contains code to create/update the database schema.
  - `InitialCreate.cs` and related files define the initial database structure.

Key Architectural Decisions

- **Separation of Concerns**: Entities, DTOs, services, and controllers are separated for maintainability.
- **DTO Usage**: Prevents object cycles and over-posting, and secures API responses.
- **Role-Based Authorization**: Ensures only Employers can manage vacancies and only Applicants can apply.
- **Background Service**: Automatically archives expired vacancies without manual intervention.
- **JWT Authentication**: Secure, stateless authentication for all API endpoints.
- **EF Core with SQL Server**: Modern, code-first ORM for robust data access.

How Everything Works Together

1. **User registers and logs in** to get a JWT token.
2. **Employers** use their token to create, update, deactivate, or delete vacancies.
3. **Applicants** search for vacancies and apply (with business rules enforced).
4. **VacancyArchiverService** runs in the background to archive expired vacancies.
5. **All data access** is via `ApplicationDbContext` and EF Core.

6. **API responses** use DTOs to avoid serialization issues and expose only necessary data.
