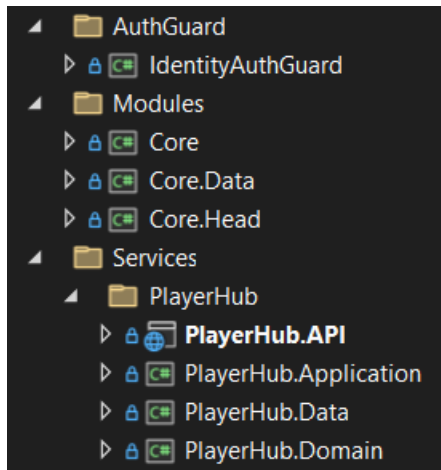


# Integrator Project

## Project Dependencies:

- .NET Core v8.0.11
- ASP.NET Core v8.0.11
- .NET Standard Library v2.1.0
- Microsoft SQL Server 2012 or later

## Structure:



**MODULES:** Contains all the requirements to start to develop any application.

- **Core** (domain layer requirements using .NET Standard Library v2.1.0): base Model, base DTO, base Domain Event.
- **Core.Data** (data access layer requirements): base DbContextFactory, base Repository, base Seed (populating a database with initial data), base UnitOfWork.
- **Core.Head** (application layer requirements): Behaviors (create pipeline behaviors to handle logging and validation to intercept and process commands, queries and events), base Command and CommandHandler, Query and QueryHandler, custom Exceptions and Global Exception Handler, APIs response Wrappers.

**AUTH GUARDS:** Provides a tailored solution for Authentication and Authorization within the application.

**AUTHGUARDIDENTITY:**

- Custom implementation of identity model mixed with custom implementation of JwtBearer.
- Provide the way to register a user, login, logout using a JwtBearer and refresh tokens.

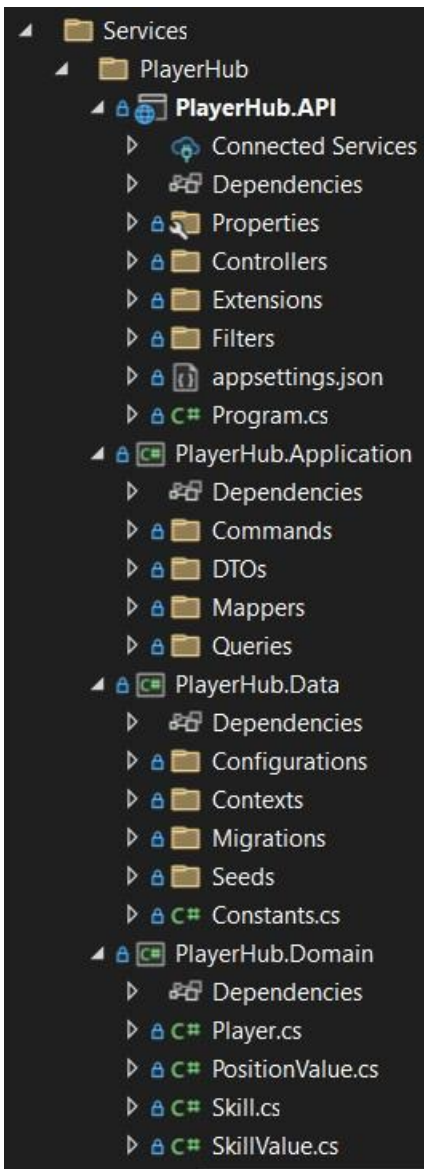
**SERVICES:** Contains the Application Services (each service is an independent project).

## PlayerHub Service:

For a deeper understanding of DDD with CQRS, refer to the following documentation: [.NET Microservices. Architecture for Containerized .NET Applications - .NET | Microsoft Learn.](#)

If you have any questions, feel free to ask me.

**DESCRIPTION:** The project is structured following the principles of Clean Architecture, incorporating a custom Domain Driven Design (DDD) and utilizing Command and Query Responsibility Segregation (CQRS), It is organized into four distinct parts:



DOMAIN LAYER: Following DDD principles, there are four types of Domain Entities: Entity, Aggregate Root, Enumeration, and Value Object, see: [Seedwork \(reusable base classes and interfaces for your domain model\) - .NET | Microsoft Learn](#).

- **.NET Standard** Library v2.1.0.
- **Domain Entities:** Player (Aggregate Root), Skill (Enumeration).

DATA LAYER:

- **Contexts:** WriteDbContext (read and write), ReadDbContext (read), AppDbContext (read and write over Identity Model).
- **Configurations:** Configure Domain Model relationships within Entity Framework Core to ensure seamless interactions and maintain database integrity.
- **Migrations:** Migrations for Domain Model and Identity Model.
- **Seeds:** Populating a database with an initial set of data.
- Although **UnitOfWork** is not implemented in this layer, its functionality is effectively managed by the data access layer. Coordinates the creation and management of repositories and the data context. It ensures atomic save operations by committing all changes in a single transaction. Additionally, it handles domain events to maintain consistency across aggregates.

### APPLICATION LAYER:

- **Commands and CommandHandlers:** Change the state of the application (creating, updating, or deleting data). Use WriteDbContext to save changes into database.
- **Queries and QueryHandlers:** Requests for data and does not alter the application's state. Use ReadDbContext to get data from database.
- **CommandValidators:** Ensure data integrity by enforcing validation rules before processing commands.
- **DTOs:** Transfer data between the client and the server.
- **Mappers:** mapping data between objects (Entity Model to DTO).

API:

- **Controllers:** Handle HTTP requests, manage CRUD operations.
  - **AccountController:** Handles account creation, login, and logout operations, and provides an endpoint for refreshing tokens.
  - **PlayerController:** Manages CRUD operations for Player entity.
- **Filters:** Enforce security policies across the application.
  - **ApiKeyAuthorizationHandler:** Validate the presence and validity of API key in the request.
- **Extensions:**
  - **ServiceCollectionExtensions:** Provides extensions for injecting services as dependencies.
  - **WebApplicationExtensions:** Offers middleware extension methods to enhance web application functionality.

## Application Request Examples:

**Note:** The x-api-key is mandatory for all client requests.

**Note:** The accessToken is required for all client requests, except for Create User and Login.

**Note:** Client requests using Postman.

**Header:**

x-api-key:AMvQopi/GgHEtDVZ80i/YxXN9LjcQEqHmcKtkuWm2+7VODWkB3JlZsOvpewwWo5Lsw==

**Access Token:** Use the access token as Bearer Token in Postman Authorization.

The image shows a screenshot of the Visual Studio Code interface. At the top, the 'Authorization' tab is selected in the top bar. Below it, the 'Auth Type' dropdown menu is open, showing 'Bearer Token' as the selected option. A warning message with an information icon states: 'Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about variables.' The 'Token' field contains a long, complex alphanumeric string. The 'Params' tab is also visible on the left.

LOGIN:

**Client Request:**

POST

{{base\_url}} /api/account/login

Send

Params

Authorization

Headers (11)

Body

Scripts

Settings

Headers

9 hidden

	Key	Value	Description		Bulk Edit	Presets
<input checked="" type="checkbox"/>	x-api-key	AMvQopi/GgHEtDVZ80l/YxXN9LJcQEgHmcKtkuWm2...				
<input type="checkbox"/>						
	Key	Value	Description			

Body

Cookies

Headers (4)

Test Results

200 OK

1.48 s

935 B

Save Response

Pretty

Raw

Preview

Visualize

JSON

```

1  {
2    "data": {
3      "accessToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJodHRwOiIvc2NoZW1hcy54bWxzbnR5Y2Zy93cy8yMDA1LzA1L2lkZW50aXR5L2NsYwltcy9uYV1awRlbnRpZmllcCI6IjE1IjA4M2YwYTVhLWM1ZmItNGU4Y5S0WzVu2LTAAZGQyY2V1NmMyNSIsImh0dHA6Ly9yY2h1bWZlbnhtbWVYXAU3bnJl3dzLzIwMDUwMDUvawRlbnRpZmllc2xhaw1zL2V2YWI5YWYWRkcmVzcyI6ImRlbnR5XpkaWVub0BnbWpCbC5jb20iL3JqdGkiOiJmMTI4OTUwM0wMTG1LTQ5M2E2MDQ0C0w0GRkMmY2ZmYmM1L3J0dHRwOiI8vc2NoZW1hcy5taWY3b3NvZnQuY29tL3dzLzIwMDgvdMDYvaW1bnRpdHkvY2xhaw1zL3JvbmV1bWZlbnhtbWZlbnhtbWVYXAU3bnJl3dzLzIwMDUwMDUvawRlbnRpZmllc2xhaw1zL2V2YWI5YWYWRkcmVzcyI6ImRlbnR5XpkaWVub0BnbWpCbC5jb20iL3JqdGkiOiJmMTI4OTUwM0wMTG1LTQ5M2E2MDQ0C0w0GRkMmY2ZmYmM1L3J0dHRwOiI8vc2NoZW1hcy5taWY3b3NvZnQuY29tL3dzLzIwMDgvdMDYvaW1bnRpdHkvY2xhaw1zL3JvbmV1bWZlbnhtbWZlbnhtbWVYXAU3bnJl3dzLzIwMDUwMDUvawRlbnRpZmllc2xhaw1zL2V2YWI5YWYWRkcmVzcyI6ImRlbnR5XpkaWVub0BnbWpCbC5jb20iL3JqdGkiOiJmMTI4OTUwM0wMTG1LTQ5M2E2MDQ0C0w0GRkMmY2ZmYmM1L3J0dHRwOiI8vc2NoZW1hcy5taWY3b3NvZnQuY29tL3dzLzIwMDgvdMDYvaW1bnRpdHkvY2xhaw1zL3JvbmV1bWZlbnhtbWZlbnhtbWVYXAU3bnJl3dzLzIwMDUwMDUvawRlbnRpZmllc2xhaw1zL2V2YWI5YWYWRkcmVzcyI6ImRlbnR5XpkaWVub0BnbWpCbC5jb20iL3JqdGkiOiJmMTI4OTUwM0wMTG1LTQ5M2E2MDQ0C0w0GRkMmY2ZmYmM1L3J0dHRwOiI8vc2NoZW1hcy5taWY3b3NvZnQuY29tL3dzLzIwMDgvdMDYvaW1bnRpdHkvY2xhaw1zL3JvbmV1bWZlbnhtbWZlbnhtbWVYXAU3bnJl3dzLzIwMDUwMDUvawRlbnRpZmllc2xhaw1zL2V2YWI5YWYWRkcmVzcyI6ImRlbnR5XpkaWVub0BnbWpCbC5jb20iL3JqdGkiOiJmMTI4OTUwM0wMTG1LTQ5M2E2MDQ0C0w0GRkMmY2ZmYmM1L3J0dHRwOiI8vc2NoZW1hcy5taWY3b3NvZnQuY29tL3dzLzIwMDgvdMDYvaW1bnRpdHkvY2xhaw1zL3JvbmV1bWZlbnhtbWZlbnhtbWVYXAU3bnJl3dzLzIwMDUwMDUvawRlbnRpZmllc2xhaw1zL2V2YWI5YWYWRkcmVzcyI6ImRlbnR5XpkaWVub0BnbWpCbC5jb20iL3JqdGkiOiJmMTI4OTUwM0wMTG1LTQ5M2E2MDQ0C0w0GRkMmY2ZmYmM1L3J0dHRwOiI8vc2NoZW1hcy5taWY3b3NvZnQuY29tL3dzLzIwMDgvdMDYvaW1bnRpdHkvY2xhaw1zL3JvbmV1bWZlbnhtbWZlbnhtbWVYXAU3bnJl3dzLzIwMDUwMDUvawRlbnRpZmllc2xhaw1zL2V2YWI5YWYWRkcmVzcyI6ImRlbnR5XpkaWVub0BnbWpCbC5jb20iL3JqdGkiOiJmMTI4OTUwM0wMTG1LTQ5M2E2MDQ0C0w0GRkMmY2ZmYmM1L3J0dHRwOiI8vc2NoZW1hcy5taWY3b3NvZnQuY29tL3dzLzIwMDgvdMDYvaW1bnRpdHkvY2xhaw1zL3JvbmV1bWZlbnhtbWZlbnhtbWVYXAU3bnJl3dzLzIwMDUwMDUvawRlbnRpZmllc2xhaw1zL2V2YWI5YWYWRkcmVzcyI6ImRlbnR5XpkaWVub0BnbWpCbC5jb20iL3JqdGkiOiJmMTI4OTUwM0wMTG1LTQ5M2E2MDQ0C0w0GRkMmY2ZmYmM1L3J0dHRwOiI8vc2NoZW1hcy5taWY3b3NvZnQuY29tL3dzLzIwMDgvdMDYvaW1bnRpdHkvY2xhaw1zL3JvbmV1bWZlbnhtbWZlbnhtbWVYXAU3bnJl3dzLzIwMDUwMDUvawRlbnRpZmllc2xhaw1zL2V2YWI5YWYWRkcmVzcyI6ImRlbnR5XpkaWVub0BnbWpCbC5jb20iL3JqdGkiOiJmMTI4OTUwM0wMTG1LTQ5M2E2MDQ0C0w0GRkMmY2ZmYmM1L3J0dHRwOiI8vc2NoZW1hcy5taWY3b3NvZnQuY29tL3dzLzIwMDgvdMDYvaW1bnRpdHkvY2xhaw1zL3JvbmV1bWZlbnhtbWZlbnhtbWVYXAU3bnJl3dzLzIwMDUwMDUvawRlbnRpZmllc2xhaw1zL2V2YWI5YWYWRkcmVzcyI6ImRlbnR5XpkaWVub0BnbWpCbC5jb20iL3JqdGkiOiJmMTI4OTUwM0wMTG1LTQ5M2E2MDQ0C0w0GRkMmY2ZmYmM1L3J0dHRwOiI8vc2NoZW1hcy5taWY3b3NvZnQuY29tL3dzLzIwMDgvdMDYvaW1bnRpdHkvY2xhaw1zL3JvbmV1bWZlbnhtbWZlbnhtbWVYXAU3bnJl3dzLzIwMDUwMDUvawRlbnRpZmllc2xhaw1zL2V2YWI5YWYWRkcmVzcyI6ImRlbnR5XpkaWVub0BnbWpCbC5jb20iL3JqdGkiOiJmMTI4OTUwM0wMTG1LTQ5M2E2MDQ0C0w0GRkMmY2ZmYmM1L3J0dHRwOiI8vc2NoZW1hcy5taWY3b3NvZnQuY29tL3dzLzIwMDgvdMDYvaW1bnRpdHkvY2xhaw1zL3JvbmV1bWZlbnhtbWZlbnhtbWVYXAU3bnJl3dzLzIwMDUwMDUvawRlbnRpZmllc2xhaw1zL2V2YWI5YWYWRkcmVzcyI6ImRlbnR5XpkaWVub0BnbWpCbC5jb20iL3JqdGkiOiJmMTI4OTUwM0wMTG1LTQ5M2E2MDQ0C0w0GRkMmY2ZmYmM1L3J0dHRwOiI8vc2NoZW1hcy5taWY3b3NvZnQuY29tL3dzLzIwMDgvdMDYvaW1bnRpdHkvY2xhaw1zL3JvbmV1bWZlbnhtbWZlbnhtbWVYXAU3bnJl3dzLzIwMDUwMDUvawRlbnRpZmllc2xhaw1zL2V2YWI5YWYWRkcmVzcyI6ImRlbnR5XpkaWVub0BnbWpCbC5jb20iL3JqdGkiOiJmMTI4OTUwM0wMTG1LTQ5M2E2MDQ0C0w0GRkMmY2ZmYmM1L3J0dHRwOiI8vc2NoZW1hcy5taWY3b3NvZnQuY29tL3dzLzIwMDgvdMDYvaW1bnRpdHkvY2xhaw1zL3JvbmV1bWZlbnhtbWZlbnhtbWVYXAU3bnJl3dzLzIwMDUwMDUvawRlbnRpZmllc2xhaw1zL2V2YWI5YWYWRkcmVzcyI6ImRlbnR5XpkaWVub0BnbWpCbC5jb20iL3JqdGkiOiJmMTI4OTUwM0wMTG1LTQ5M2E2MDQ0C0w0GRkMmY2ZmYmM1L3J0dHRwOiI8vc2NoZW1hcy5taWY3b3NvZnQuY29tL3dzLzIwMDgvdMDYvaW1bnRpdHkvY2xhaw1zL3JvbmV1bWZlbnhtbWZlbnhtbWVYXAU3bnJl3dzLzIwMDUwMDUvawRlbnRpZmllc2xhaw1zL2V2YWI5YWYWRkcmVzcyI6ImRlbnR5XpkaWVub0BnbWpCbC5jb20iL3JqdGkiOiJmMTI4OTUwM0wMTG1LTQ5M2E2MDQ0C0w0GRkMmY2ZmYmM1L3J0dHRwOiI8vc2NoZW1hcy5taWY3b3NvZnQuY29tL3dzLzIwMDgvdMDYvaW1bnRpdHkvY2xhaw1zL3JvbmV1bWZlbnhtbWZlbnhtbWVYXAU3bnJl3dzLzIwMDUwMDUvawRlbnRpZmllc2xhaw1zL2V2YWI5YWYWRkcmVzcyI6ImRlbnR5XpkaWVub0BnbWpCbC5jb20iL3JqdGkiOiJmMTI4OTUwM0wMTG1LTQ5M2E2MDQ0C0w0GRkMmY2ZmYmM1L3J0dHRwOiI8vc2NoZW1hcy5taWY3b3NvZnQuY29tL3dzLzIwMDgvdMDYvaW1bnRpdHkvY2xhaw1zL3JvbmV1bWZlbnhtbWZlbnhtbWVYXAU3bnJl3dzLzIwMDUwMDUvawRlbnRpZmllc2xhaw1zL2V2YWI5YWYWRkcmVzcyI6ImRlbnR5XpkaWVub0BnbWpCbC5jb20iL3JqdGkiOiJmMTI4OTUwM0wMTG1LTQ5M2E2MDQ0C0w0GRkMmY2ZmYmM1L3J0dHRwOiI8vc2NoZW1hcy5taWY3b3NvZnQuY29tL3dzLzIwMDgvdMDYvaW1
```

**Body:**

```
{
  "email": "dj.diazdiego@gmail.com",
  "password": "Set_Your_Password_123"
}
```

The above credentials belong to a default Admin user created through data seeding.

## CREATE A PLAYER:

**Client Request:**

**POST** ▼ `{{base_url}}/api/player`

Params   Authorization   Headers (11)   **Body** ●   Scripts ●   Settings

☐ none   ☐ form-data   ☐ x-www-form-urlencoded   ☒ raw   ☐ binary   ☐ GraphQL   **JSON** ▼

```
1 {
2   "name": "john",
3   "position": "forward",
4   "skills": [
5     "attack",
6     "speed",
7     "strength"
8   ]
9 }
```

GET PLAYERS:

### Client Request:

The screenshot shows a REST client interface with a GET request to `{{base_url}} /api/player?$expand=Skills&$orderby=Name desc, Position asc&$top=10&$skip=0&$filter=position eq 'defender' or position eq 'midfielder'`. The query parameters are listed as follows:

```
$expand:Skills
$orderby:Name desc, Position asc
$top:10
$skip:0
$filter:position eq 'defender' or position eq 'midfielder'
$search:ged
```

The status bar at the bottom indicates a 200 OK response with 88 ms latency and 929 B of data.

**Note:** The endpoint utilizes OData to construct queries. For more information, see: [OData overview - OData | Microsoft Learn](#) and [Query options overview - OData | Microsoft Learn](#). For a comprehensive understanding of OData, refer to the following documentation: [OData documentation - OData | Microsoft Learn](#).

### Application Basic Command Flow (Start in Client App):

